Towards a systematic security evaluation of the automotive Bluetooth interface

Cheah, Hun

Award date:
2018

Awarding institution:
Coventry University

TOWARDS A SYSTEMATIC SECURITY EVALUATION OF THE AUTOMOTIVE BLUETOOTH INTERFACE

HUN XHING (MADELINE) CHEAH

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Centre for Mobility and Transport Research
Coventry University

in collaboration with HORIBA MIRA

October 2017

Hun Xhing (Madeline) Cheah: *Towards a Systematic Security Evaluation of the Automotive Bluetooth Interface, October 2017*

Certificate of Ethical Approval

Applicant:

Hun Cheah

Project Title:

Towards a systematic security evaluation of the automotive Bluetooth interface

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

19 June 2017

Project Reference Number:

P45211

AUTHOR

Hun Xhing (Madeline) Cheah
Email: *cheahh2@uni.coventry.ac.uk*

SUPERVISORS

Dr. Siraj A. Shaikh: *Professor of Systems Security, Coventry University*
Dr. Olivier Haas: *Reader in Applied Control Systems, Coventry University*
Dr. Alastair Ruddle: *Computational Electromagnetics Consultant, HORIBA MIRA*
Dr. Jeremy Bryans: *Research Fellow, Coventry University*

DESIGN

The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final Version as of January 17, 2018 (`classicthesis`).

Take me with you.
For laughs, for luck, for the unknown.

— Peter S. Beagle

Dedicated to my family

ABSTRACT

In-cabin connectivity and its enabling technologies have increased dramatically in recent years. Security was not considered an essential property, a mind-set that has shifted significantly due to the appearance of demonstrated vulnerabilities in these connected vehicles. Connectivity allows the possibility that an external attacker may compromise the security - and therefore the safety - of the vehicle. Many exploits have already been demonstrated in literature. One of the most pervasive connective technologies is Bluetooth, a short-range wireless communication technology. Security issues with this technology are well-documented, albeit in other domains. A threat intelligence study was carried out to substantiate this motivation and finds that while the general trend is towards increasing (relative) security in automotive Bluetooth implementations, there is still significant technological lag when compared to more traditional computing systems.

The main contribution of this thesis is a framework for the systematic security evaluation of the automotive Bluetooth interface from a black-box perspective (as technical specifications were loose or absent). Tests were performed through both the vehicle's native connection and through Bluetooth-enabled aftermarket devices attached to the vehicle. This framework is supported through the use of attack trees and principles as outlined in the Penetration Testing Execution Standard. Furthermore, a proof-of-concept tool was developed to implement this framework in a semi-automated manner, to carry out testing on real-world vehicles. The tool also allows for severity classification of the results acquired, as outlined in the SAE J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. Results of the severity classification are validated through domain expert review. Finally, how formal methods could be integrated into the framework and tool to improve confidence and rigour, and to demonstrate how future iterations of design could be improved is also explored.

In conclusion, there is a need for systematic security testing, based on the findings of the threat intelligence study. The systematic evaluation and the developed tool successfully found weaknesses in both the automotive Bluetooth interface and in the vehicle itself through Bluetooth-enabled aftermarket devices. Furthermore, the results of applying this framework provide a focus for counter-measure development and could be used as evidence in a security assurance case. The systematic evaluation framework also allows for formal methods to be introduced for added rigour and confidence. Demonstrations of how this might be performed (with case studies) were presented. Future recommendations include using this framework with more test vehicles and expanding on the existing attack trees that form the heart of the evaluation. Further work on the tool chain would also be desirable. This would enable further accuracy of any testing or modelling required, and would also take automation of the entire process further.

PUBLICATIONS

Presented here are publications where work that has been carried out has either informed or been included in the thesis.

M. Cheah, H.N. Nguyen, J. Bryans, S. Shaikh, "Formalising Systematic Security Evaluations using Attack Trees for Automotive Applications" *in 11th International Conference on Information Security Theory and Practice (WISTP)*, Crete: Springer, Sep 2017

M. Cheah, S. Shaikh, O. Haas, A. Ruddle, "Towards a systematic security evaluation of the automotive Bluetooth interface" *Journal of Vehicular Communications*, vol. 9, July, pp. 8-18, 2017

M. Cheah, J. Bryans, D. Fowler, S. Shaikh, "Threat Intelligence for Bluetooth-enabled Systems with Automotive Applications: An Empirical Study" in *47th IEEE/IFIP Dependable Systems and Networks Workshops: Security and Safety in Vehicles (SSIV)*, Denver: IEEE, June 2017

D. Fowler, **M. Cheah**, S. Shaikh, J.Bryans, "Towards A Testbed for Automotive Security" in *10th IEEE International Conference on Software Testing, Verification and Validation (ICST): Industry Track*, Tokyo: IEEE, March 2017 (Best Industry Track Paper Award)

M. Cheah, S. Shaikh, J. Bryans, H.N. Nguyen, "Combining third party components securely in automotive manufacture" *in 10th International Conference on Information Security Theory and Practice (WISTP)*, Crete: Springer, Sep 2016

M. Cheah, S. Shaikh, J. Bryans, H.N. Nguyen, "Design and testing methodology for automotive security" in *International Symposium on Engineering Secure Software and Systems: Doctoral Symposium*, London, Apr 2016

M. Cheah and S. Shaikh, "Autonomous Vehicle Security," *IET Engineering and Technology Reference*, vol. 1, no. 1, 2015.

I was taught that the way of progress
was neither swift nor easy.
— Marie Curie

ACKNOWLEDGMENTS

I would like to especially thank Prof. Siraj Shaikh for guiding this research, for pushing me by setting (hardcore) deadlines, for the opportunities gratefully received and, generally, for the incredible amount of support that has come my way. I would also like to thank Dr. Jeremy Bryans for extremely stimulating conversations, for his always on-point advice and for consistently reminding me that I should be concise (I hope this is enough?). Thanks also goes to Dr. Hoang Nga Nguyen for his advice and ideas around how this research could be taken into the future and for his help and explanations regarding formal methods, and Dr. Olivier Haas for valuable comments and guidance throughout this programme.

Additionally, this work would not have been possible without the contribution and support of HORIBA MIRA, and especial thanks goes to Mr. Paul Wooderson, Dr. Anthony Baxendale and Dr. Alastair Ruddle for their assistance, guidance and interest in this project. Thanks are also due to Paul and Mr. Anthony Jude for agreeing to validate (as domain experts) the work that I've done. I would also like to acknowledge, with gratitude, the financial assistance provided by Coventry University and HORIBA MIRA, without which this work would not have been completed on time.

A big thank you goes to my partner Alex Roache for putting up with me on this excruciatingly intense journey (and for the fantastically made cups of tea). Last, but absolutely and certainly not least, thank you to all my family, who have been the consummate cheerleaders from afar.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AUTOMOTIVE

ASEAL Automotive Security Assurance Level: *Proposed (but not yet adopted) classification to build security assurance cases for automotive systems*

CAN Controller Area Network: *Primary communications protocol in intra-vehicular networks*

ECU Electronic Controller Unit: *Small unit typically containing a microprocessor, used to control various functionality on the vehicle*

OBD-II On-board diagnostic II port: *Used for diagnostics and maintenance. Legally mandated*

OEM Original Equipment Manufacturer: *Refers to vehicle manufacturers, but typically means a company that creates or supplies components that are sold under different branding*

OTS Off-the-shelf: *Term used to describe components bought from third party suppliers as a whole unit or sub-unit*

PID Parameter ID: *A form of diagnostic message to be sent into the OBD-II port*

PKES Passive Key Entry and Start: *A technology based on radio frequency used to unlock and start the vehicle as long as correct key is within range*

SID Service ID: *Have specified pre-determined functionality, that can be sent to ECUs for diagnostics. Part of Unified Diagnostic Services*

SoC System-on-a-chip: *Chip containing the operating system*

UDS Unified Diagnostic Services: *A standard specifying the format (and some functionality) of diagnostic messages*

VIN Vehicle Identification Number: *A number that is unique to each vehicle, used for maintenance and recovery of stolen vehicles*

SECURITY AND COMPUTING

AT "Attention modem": *A set of commands used to configure a serial device*

ATI AT Information: *Command to request device information*

ATMA AT Monitor All: *Command used to monitor all traffic on the CAN bus through the OBD-II port*

ATSP AT Set Protocol: *Command to set protocol to the specific CAN protocol in use through the OBD-II port*

CIA Confidentiality, Integrity, Availability: *The three basic tenets of security, usually represented as a triangle*

CRC Cyclic redundancy check: *Used for error detection*

CSP Communicating Sequential Processes: *A formal language used to describe communications and interaction in systems*

CSR Cambridge Silicon Radio: *A manufacturer of, amongst other things, Bluetooth chipsets*

CSV Comma-separated values: *A file format used to store data*

CVE Common Vulnerabilities Enumeration: *A database of vulnerabilities found in specific implementations*

CWE Common Weaknesses Enumeration: *A database of weaknesses found in software*

DCE Data Communication Equipment: *Device or instrument used to establish, maintain, transmit or receive communications*

DFD Data Flow Diagrams: *Visualisation method for threat modelling*

DoS Denial of Service: *Any tool or method that violates the availability requirement of security*

DTE Data Terminal Equipment: *A destination device that converts user information into signals or vice versa. Communicates with DCE*

DREAD Damage potential, Reproducibility, Exploitability, Affected users, Discoverability: *A popular methodology for threat modelling*

FDR Failures-Divergences Refinement: *A tool for refinement checking*

FUSE Filesystem in Userspace: *Software interface for creating userspace filesystems without editing kernel code*

GPS Global Positioning System: *Used for satellite navigation*

IDS Intrusion Detection System: *Technology used to detect any anomalous behaviour (that could be indicative of an attack) on a system*

MAC Media Access Control (address) *Globally unique machine identifier*

MBST Model-based security testing: *Security testing using formal models*

MITM Man-in-the-middle: *A form of attack where an adversary sits in the middle of a communication to eavesdrop or inject*

OOB Out-of-Band: *An association model used by SSP, where authentication uses other technologies such as Near Field Communications*

OS Operating system: *Software platform on which higher-level applications sit*

PIN Personal Identification Number: *Used in many authentication processes*

PTES Penetration Testing Execution Standard: *A set of technical guidelines for penetration testing*

RSSI Received Signal Strength Indicator: *A measure of signal strength received from an access point*

SAND Sequential AND: *A logic gate that dictates that all leaf nodes must be completed in temporal order*

SP Series-parallel (Graph): *A directed formal structure which can be used for testing and proofs*

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege: *A popular methodology for threat modelling*

SQUARE System Quality Requirements Engineering: *A methodology for requirements gathering and actioning*

USB Universal Serial Bus: *Port used for data exchange*

BLUETOOTH

A2DP Advanced Audio Distribution Profile: *Encodes procedures for distribution of high quality audio*

AVRCP Audio/Video Remote Control Profile: *Provides remote control functionality to audio or video equipment*

BDADDR Bluetooth Address: *The unique identifier for each Bluetooth-enabled device*

Bluetooth SIG Bluetooth Special Interest Group: *The organisation that manages the Bluetooth technology*

BNEP Bluetooth Network Encapsulation Protocol: *Encapsulates Ethernet packets for exchange via the Personal Ad-hoc Network profile*

BRSF Bluetooth Retrieve Supported Features: *Used to report on features supported by a hands-free device*

CSR Cambridge Silicon Radio: *A company that creates Bluetooth chipsets*

FHSS Frequency Hopping Spread Spectrum: *A mechanism used by Bluetooth, designed to reduce or avoid interference*

FTP File Transfer Profile: *Provides the ability to transfer files between devices*

HCI Host Controller Interface: *Command interface to the Bluetooth baseband controller*

HFP Hands Free Profile: *Allows for voice connections and remote control of a connected device*

HSP Headset Profile: *Allows for audio between two devices*

L2CAP Logical Link Control and Adaptation Protocol: *Used to transport protocols such as RFCOMM to higher level protocols*

LAP Lower address part: *The last 24 bits of a Bluetooth address*

LMP Link Manager Protocol: *Handles low-level negotiations in Bluetooth*

MAP Message access profile: *Provides ability to exchange messages between devices*

MNS Message Notification Service: *Works in conjunction with MAP to allow for notification of receipt of messages*

MTU Maximum Transmission Unit: *Measure of the largest packet that can be sent through an L2CAP port (in bytes)*

NAP Non-significant address part: *The first 16 bits of the Bluetooth address*

NINO NoInput-NoOutput: *A specific attack on the capabilities exchange phase in Bluetooth, allowing for the weakest association model to be used when pairing*

OBEX Object Exchange Protocol: *Communications protocol in Bluetooth that allows for object exchange between connected devices*

OPP Object Push Profile: *Provides ability to push messages or data onto a connected Bluetooth-enabled device*

OUI Organisationally Unique Identifier: *The first 24 bits of the Bluetooth address, which correspond to a manufacturer registered to Bluetooth SIG*

PANU Personal Ad-hoc Network User profile: *Provides ability to receive Ethernet packets*

PBAP Phonebook Access Profile: *Provides access to phonebook objects such as contacts*

RFCOMM Radio Frequency Communications: *Provides emulation of serial ports*

SDP Service Discovery Protocol: *Used to discover service profiles on offer by a Bluetooth-enabled device*

SPP Serial Port Profile: *Provides the ability to use a serial connection*

SSP Secure Simple Pairing: *Pairing mechanism used by Bluetooth, employing Elliptic Curve Diffie-Hellman public-key cryptography*

SyncML Synchronization Markup Language: *Former name for a synchronisation standard, now known as "Open Mobile Alliance Data Synchronization and Management"*

UAP Upper address part: *The third byte of the Bluetooth address*

STANDARDS AND ORGANISATIONS

EVITA E-safety Vehicle Intrusion Protected Applications: *A project aimed at protecting vehicular systems from intrusion by providing a secure architecture for on-board automotive networks*

FCC Federal Communications Commission: *A body in the US responsible for licensing any component or system that involves radio communications*

HEAVENS Healing Vulnerabilities to Enhance Software Security: *A project aimed at protecting vehicular software*

ISAC Information Sharing and Analysis Centres: *Organisations dedicated to information sharing within a certain industry*

ISO International Standards Organisation: *A body that sets global standards*

SAE Society of Automotive Engineers: *A professional association for engineers*

Part I

THE THREAT LANDSCAPE

"If you have built castles in the air, your work need not be lost; that is where they should be. Now put the foundations under them."

Henry David Thoreau

1

INTRODUCTION

Historically, embedded systems were designed to operate in a tightly-controlled environment which required specialist knowledge to design, calibrate and deploy. Security was, at best, only superficially considered during this period [96]. This mind-set began to shift when the increasing number of microprocessors and complexity of software, increase in functionality and growth of external-facing surfaces became apparent [63], [75].

There are several major developments which have contributed to the automotive threat landscape:

- Firstly, the presence of increased amounts of software (i.e. lines of code) to deal with increasingly sophisticated functionality and attendant rise in the number of processing units. This leads to compounded complexity. Subsequently, testability (and in this context, security testing) could become impaired and the likelihood of large numbers and severity of vulnerabilities increases [125].

- Secondly, there has been significant development and integration of (wireless) communication interfaces, which means more connectivity. This has led to increased number of connections in the intra-vehicular network as reuse of externally provided information becomes more important. There is also a concomitant rise in the number of external peripheral devices that can now connect to the vehicle. This means that there are now more access points for malicious attackers, which also potentially negatively impacts system boundaries by blurring them, or extending them to beyond the control of original manufacturers such that unknown interactions (and therefore possible security risks) could exist.

- Finally, content volume, variability and value has changed and increased, which means that there is more data about the ve-

hicle to extract, with more potential value which could be extracted if personal data is also stored on the vehicle.

1.1 MOTIVATION

The consensus is that security engineering (and security testing as part of that process) is still relatively novel in mainstream automotive production [10], [137], and that security is incidental and usually a by-product of achieving performance and safety goals [36], [63], [91], [128]. Even with advanced formal methods for modelling and testing, the need for and number of demands, features and increased connecting power means that, even had security been considered, the scale of the problem facing security testers is now much broader [12].

There are several challenges to securing interfaces in vehicles. Any security mechanism will require additional processing overhead, and on the hardware level, has ramifications in provision of energy and in physical assembly and design, such as placement of additional wiring. Even should such concerns be addressed, countermeasures that are commonly used currently for large and complex systems are not suitable for vehicular embedded systems because of hardware constraints and the differences in network configuration. Well-established defences at software level such as the use of cryptography, firewalls and intrusion detection systems (IDS) cannot be implemented without considerable change in architecture due to the use of sufficiently different protocols and topologies within the automotive domain. Post-release, maintenance becomes an issue as patches for discovered vulnerabilities, unless performed over-the-air, are difficult to apply once units are sold.

Although the vast majority of demonstrated attacks that have been directed at the vehicle use automotive-specific vectors, many of the methods are familiar to security professionals. This includes the use of malware and known software vulnerabilities, proximity extending hardware, replay attacks or simply reverse engineering to gain illicit knowledge of the system [36]. Considering the similarity of attack methods, parallels can be drawn between non-automotive and automotive systems, forming a baseline from which to draw information on possible weaknesses.

All of the above is dependent on acquiring knowledge and information regarding existing vulnerabilities. From the number and variety

of reported threats, weaknesses and exploits (see Chapter 2.1.1), it is clear that a methodical description of the problem - a systematic security evaluation - is essential.

The investigations in this thesis centre around the implementation of Bluetooth technology on vehicles. This is because Bluetooth is a pervasive interface and was therefore chosen for study because of the potential negative impact should it be compromised. There have been estimates that vehicles with a Bluetooth interface number at nearly nine million currently, with a forecast of 21 million vehicles to have Bluetooth by 2018 [57]. Market growth for wireless systems, of which Bluetooth is a major enabler, is anticipated to grow by more than 40% between 2012 and 2018 with market revenue set to rise to $1.6 billion in 2018 [78].

1.2 RESEARCH QUESTIONS

A systematic security evaluation method has many advantages. There is a disparity between what an attacker must find in order to exploit the system (potentially just one vulnerability) and the number of flaws a defender would have to safeguard in order to protect the system (as many as possible). An ad-hoc approach to finding vulnerabilities - which by implication means a subjective prioritisation of what and where to test [100] - potentially results in flaws being overlooked. A methodical approach increases the likelihood of determining flaws, thereby mitigating this problem [138]. This is even more important since information sharing is still limited due to the competitive nature of the industry [102]. Systematic analyses can also be supported by a variety of tools. An advantage of this is that the final result can be documented, with all the details that led to the system compromise [41].

Considering the above, this doctoral research addresses two questions:

- Firstly, how do we systematically establish a baseline security state of the system when original specifications are absent? and
- Secondly, once the problem has been enumerated, how do we use the results of such testing to aid in security assurance and in future iterations of design?

The research questions posed are in the context of Bluetooth implementation natively available through the vehicle's information and entertainment system and through any Bluetooth-enabled devices that are deployed in the vehicle retroactively.

1.3 CONTRIBUTIONS

The main contributions of this thesis are:

- **Threat intelligence** regarding the presence of publicly broadcasting Bluetooth-enabled devices (both through the vehicle head-unit and through attached aftermarket devices) in a real-world scenario. This also contributes to situational awareness regarding the technological lag between year of vehicle registration and year that versions of Bluetooth technology was adopted;
- **Systematic security evaluation** of the automotive Bluetooth interface, implemented in a proof-of-concept tool. This tool also contributes towards the automation of the security evaluation;
- **Classification of evidence** acquired from the security testing process, using an industry standard severity classification scheme. This encompasses two aspects, privacy and operational, and the results can be used as evidence in a security assurance case;
- **Conceptual methodology** demonstrating how both the processes and the results of empirical testing can be further formalised to confer additional rigour and confidence regarding future testing and design iterations.

1.4 THESIS STRUCTURE

The rest of the thesis is organised as follows:

Chapter 2 presents an overview of experimental analyses that have resulted in the discovery of the nature of vulnerabilities in vehicles, as well as the state of the art around comparable methodologies. This includes automotive specific evaluations as described in emerging standards.

Chapter 3 focuses on Bluetooth as a technology, providing an overview of the protocol, the known security issues and existing tooling for exploring known vulnerabilities, albeit not in the automotive domain. The contribution in this chapter is a threat intelligence study regarding the use of Bluetooth as implemented or used in vehicles.

Chapter 4 describes and explores the methodology used for the systematic evaluation of the automotive Bluetooth interface. The purpose of this chapter is to present the systematic framework developed, supported by attack trees and penetration testing.

Chapter 5 discusses the implementation and validation of the proof-of-concept tool developed in accordance with the methodology as specified in Chapter 4. The contribution here is the (semi) automation of the process by which an automotive interface can be systematically evaluated from a security perspective.

In Chapter 6, the experimental analysis of Bluetooth as implemented in automotive information and entertainment headunits is presented, along with severity classifications of the findings. The contribution here of evidence classification has been evaluated by domain experts.

Chapter 7 focuses on experimental results and the implications thereof in aftermarket Bluetooth-enabled devices, specifically looking at aftermarket devices that attach to a vehicle's diagnostics port.

Chapter 8 looks at addressing the challenge of introducing formal methods (for additional confidence and robustness) when testing a vehicular black box. This is explored on two fronts: firstly, through formal analysis and secondly, by formalising the attack trees used in the systematic security evaluation in preceding chapters.

Finally, concluding remarks, a summary of contributions and future directions are given in Chapter 9.

2

RELATED WORK

This chapter presents related work in the field of automotive cybersecurity. First, demonstrated exploits (which encompasses most of the early work in the area) are discussed (Section 2.1.1). The revelation of these exploits made it clear that a structured approach was needed to cover all the security-related aspects of development and testing. Comparative methods for this are explored in Section 2.1.2.

There is a marked dichotomy in the literature; a large body of papers explore or propose security engineering methods and tools, whilst another body describe and discuss methods by which the exploits were exposed. The former used formal methods in many cases, whilst the latter was dependent on real-world implementation tests. Standards bodies have therefore begun to look at integrating these two streams in a full design, development, implementation and testing process in standards such as J3061, and these are discussed in Section 2.2.

2.1 SECURITY TESTING IN THE AUTOMOTIVE DOMAIN

A feature of security, in any practice whether physical, digital or any combination thereof, can be described as a series of foot-races. Advances in technology with the aim of implementing greater security inspires ever more sophisticated attacks, which subsequently leads to more technological development, which is again circumvented in a yet more advanced manner. It should also be recognised that security is not a "set of features" but rather is a property of a system [71]. It is not enough to implement a strong cryptographic system, if it can be easily bypassed by poor operating practices. Furthermore, like safety or efficiency, security should be a cross-section consideration.

The *cyber*security aspect of testing concerns only software and its interactions. It should not be confused with the testing of physical security (such as door locks, or car alarms) although the two may overlap, especially if physical security features have been enabled or enhanced by software.

2.1.1 *Automotive Cybersecurity*

There have been many studies investigating software weaknesses in the vehicle. An example includes finding vulnerabilities in the software that governs a CD player (and the unit that it is attached to). By encoding arbitrary Controller Area Network (CAN) packets in a music format, Checkoway, McCoy, Kantor, *et al.* [36] was able to cause a buffer overflow. Using a later development they were also able to reflash the media electronic control unit (ECU). Other demonstrations include brute forcing the native default WiFi Personal Identification Number (PIN) to disable car alarms [101]. These have shown that there are vehicles present on the road with potentially mission-critical systems that were not sufficiently protected. This lack of protection has also led to the ability to compromise these systems through the on-board diagnostics (OBD) port [93], or by otherwise tapping into the intra-vehicular Controller Area Network (CAN) bus [38], [73] through other means.

Of most interest in this thesis were wireless or remote attacks. These present the greatest risk, since a physical in-cabin presence is unnecessary. This has also been demonstrated in literature. Some examples include injecting malware down the in-cabin phone, or via a compromised Android device connected to the vehicle [36] using a malicious self-diagnostic app to compromise the OBD-II port through Bluetooth [161]. Alternatively, another study has discussed being able to take advantage of easily guessable Bluetooth PINs to inject audio files and eavesdrop on in-cabin conversations (implemented in a tool called CarWhisperer) [167].

Keys were also a target, with relay attacks detailed in [54]. This allowed the typical short distance of key to vehicle communication to be amplified, with every passive key entry and start (PKES) system susceptible. Encryption protocols governing keys and immobilisers in the vehicle have also been demonstrably compromised [152], [153] since embedded system constraints meant that very small encryption keys were used.

Packets (and their unique fixed identifiers) could be captured from tyre pressure monitoring systems [132] from as far away as 40m. Because of the length of the identifier field, there is sufficient uniqueness to enable tracking of vehicles and their respective locations. This could potentially compromise privacy.

There are also exploits that have been reported through the media or through "hacker conferences". The exploitation of the cellular and WiFi interfaces famously led to compromise of a vehicle's control systems over the Internet [110]. Others have been able to compromise an aftermarket dongle [7] to then compromise the vehicle through use of messages originally meant for diagnostics. Laboratory exercises, where a digital audio broadcast (DAB) station was created to send data via DAB signals, have thrown up the possibility that text or images could be injected into an infotainment unit [151]. However, there has not yet been a demonstration to show how this might affect mission critical systems.

Although the papers detailed above show an impressive range of experimentation and an in-depth knowledge of the target system or component, they have not mapped out a process or systematised their findings. Furthermore, information on the practical aspect of automotive security testing is scarce: automotive systems are complex with many different technologies integrated into the single vehicle, and thus many papers dealing with experimental analysis, by necessity, limit their scope to a single interface, protocol or technology. However, there are points of agreement on both actual and potential attack vectors. For example, many agree that Bluetooth is a viable entry point for an attacker [42], [73], [87], [107], [120], [159]. Nevertheless, despite the paucity of information, from the number and variety of reported threats, vulnerabilities and exploits, it is clear that a systematic description of the problem is required.

2.1.2 *Comparative Methods*

Not unlike the software in embedded systems, conventional computing software exists in an environment where there is a significant level of co-dependence, whether it be the loading of libraries, or interfacing with third party components. This is analogous to vehicular embedded systems having nodes (ECUs) of various functions which are required to interact with equipment, firmware and possibly high-level software (in the case of infotainment systems) developed by third parties. Dependencies in all of these cases means that there are two issues that may need to be considered: firstly, that the system may inherit weaknesses from one or more components that it depends upon,

and secondly that any external security measure (as part of the larger system boundary) might also fail.

Due to architectural and implementation heterogeneity and complexity (especially in a vehicle), there may also be dependencies that are unforeseen or undocumented when components are integrated into the larger system [136]. Furthermore, due to the common practice of reuse in the automotive sector [125], weaknesses could be passed from one generation of vehicle to the next, or laterally across many makes and models depending on the supplier or manufacturer.

Because we can draw analogies between traditional high-level software and the software that exists in vehicles, we can use certain methods as a basis for the cybersecurity testing of a vehicle.

There are established security testing techniques and methods that could be drawn from previous work and applied to the automotive context. These are explored below.

2.1.2.1 *Model-based Security Testing*

Security testing is inherently hard [51], [147], due in part to the fact that no kind of testing can show the absence of security flaws. Furthermore, security testing can also be unstructured, non-reproducible, undocumented or untraceable [51]. This is the primary motivation for creating explicit models that contain information about the system under test and its environment. These test models can be built from requirements or specifications that already exist and its interactions, properties and behaviours are usually expressed in some formal language, such as the Communicating Sequential Processes (CSP) process algebra.

Systematic evaluations have been described in model-based testing studies such as that of [105], where test case generation starts from a set of models, which are then executed on a system under test, using a tool chain to stimulate the system. Reactions are then observed and evaluated. The security perspective of this method is employed in model-based security testing (MBST) [138]. MBST (which is an active field of research) usually focuses on violation of security properties, which are also formal statements in mathematical logic expressed in conjunction with a formal system specification.

Model-based security testing may provide coverage of many kinds (structural, data, requirements etc.) pertaining to security of or in a system. However, applications thereof [86] have required that models

be available or pre-built in order to formally examine. A generic system model could be created, such as in the work done by [136], where a system model of the automotive on-board network and an attacker model were created. However, at that level of abstraction, security flaws are difficult to test for, since many are based on flaws in specification of the exact system or in implementation [81] or are lateral (i.e. based on unintended functionality). Formal threat models created, such as that of [137], yields many benefits such as precision, in-depth understanding of the attack in question and automatic test-case generation. However, it would still require that the exact pathways of the attack through a system and its environment be known.

The previous works as described above has inspired the systematic approach used in this thesis, in particular the use of attack trees and subsequent testing based on a proof-of-concept tool chain (see Chapter 5). However, although model-based testing approaches provide rigour and confidence, there is no trustworthy model from which to generate tests. Even where specifications are available, the environment is sufficiently different such that functional accuracy is difficult. For example, the Bluetooth specifications (which are freely available) differ based on the version, some of which are substantially changed from one version to the next (such as the difference between legacy pairing and Secure Simple Pairing from Bluetooth 2.0 to Bluetooth 2.1 - see Chapter 3). Furthermore, the Bluetooth implementation would be integrated with other systems (such as the embedded system's operating system and other firmware) for which we would need to include to provide a complete model representation of the system and for which there is very little information.

To conclude, the primary barrier to using such methods is that the information required to do so is not available, both due to commercial confidentiality and the obscurity of sub-components within the system (many of which are third party). This also precludes other methods of enabling systematic evaluation such as attack graphs, for which formal model checking could be performed.

2.1.2.2 *Test case generation*

Because test case generation from models is not feasible in this study (see above), approaches from requirements engineering could be used instead.

A use case would usually help define functional requirements (i.e. what a piece of software or a system *should* do), which can be tested against. However, it is the intentional misuse (i.e. abuse) cases that can help identify security requirements [71], [163] by first elucidating the possible actions of an adversary, testing these actions and acting on the results to create countermeasures. In this way, security requirements can be mapped to components, systems or system interactions, ensuring that flaws in design are minimised. However, security requirements differ in paradigm to functional requirements in that usually they are about protection and bounding of a system (i.e. not allowing something to happen, rather than making sure something does happen) which is difficult without some kind of problem definition (the abuse case).

Abuse cases that have already been realised in analogous domains could be used as a template from which to test the system (in this case the vehicle). Inference of abuse cases to test would also highlight the absence of security requirements, which may also give insight into the entry points and vulnerabilities of the system. This is by necessity based on a threat model, which can be both formal or informal (see Chapter 4). This is then used to design the test cases and is a process that is used in this thesis in order to generate the attack tree (see Chapter 4).

Similar methods for gathering security requirements have been proposed by Fuchs and Rieke [56], who emphasise the criticality of prioritising security requirements in early design phases (although we deal with an already implemented system in this thesis — this is discussed in Chapter 4). Advantages to this approach include the fact that a "systems of systems" perspective is taken, and is similar to our approach in that security requirements are linked to possible attacks. A key difference to the methodology as presented here, however, is that the methods described by Fuchs and Rieke [56] are predicated again on the functional model of the system (based on "atomic actions" and functional dependencies) which results in the same issue as before: that there is no information readily available.

There have also been comparative approaches specifically using attack trees (Section 4.1) in a requirements gathering and actioning process. One of these methods is the System Quality Requirements Engineering (SQUARE) methodology [62]. However, use cases in this methodology concentrated on security threats related to a company's

technical and operational procedures rather than embedded systems (in particular the automotive system), which require consideration of manufacturing processes and constraints.

Another related approach is the formation of "anti-models" [95], depicting how model elements may be compromised or threatened (analogous to attack trees). However, these anti-models are derived from the model of the system-to-be (with attendant high informational needs), which makes it less suitable for a system with unknown internals. Even where there are methods that allow for only partial specifications (such as the framework based on Model Driven Engineering) [81], perfectly legitimate functional behaviour in those specifications could actually be a weakness in terms of the larger system boundary (see Chapter 8), for which there may be no information.

2.1.2.3 *Attack patterns*

Once test case generation has been addressed to a satisfactory degree, stimulation of the system is required to actualise some data. As part of this testing process, techniques that have been used in the past to expose vulnerabilities in analogous domains could be used as starting points to probe an unknown system.

Although not intentional, many vulnerabilities are actually designed into an application. For example, test instrumentation - where programme interfaces are added for testing purposes - are sometimes not closed or resolved. Ports could be left open or unsecured, or default configurations could be weak or contradictory. The diagnostics port on a vehicle is a prime example; it made vehicular diagnostics far simpler, but also exposed internal networks to the outside world, potentially without any gateway to filter or otherwise act on the receipt of malicious messages. This could lead to a whole spectrum of problems, ranging from monetary loss (i.e. theft) to actual physical harm should the vehicle's controls be significantly impaired.

Other techniques include side channel attacks such as timing attacks to deduce cryptographic keys as demonstrated by [91], as well as fuzzing which could be used to compromise the vehicle, as demonstrated by [93]. Unanticipated user input such as reserved words, escape characters, long strings or boundary values could all cause problems. The modern vehicle, especially with modern infotainment systems, is not immune to this: a study [36] has demonstrated how Bluetooth and cellular wireless technologies can be exploited to gain

complete control of the automobile using buffer overflows in order to leverage an authentication weakness.

This relatively small number of techniques forms the basis for many of the popular attacks seen against services and software (such as buffer overruns) [60]. These have since been encoded in *attack patterns* [52], a taxonomy based approach which contains set descriptions of the vulnerabilities found in a system. These descriptions largely comprise information regarding the vulnerability's location, targets, context and countermeasures, and can be created based on an attacker's point of view [52]. Note that there needs to be sufficient detail to form these patterns [60] and that, even if the theoretical attacks are determined, practical attacks can still fail due to lack of entry points, information or testing opportunities [52]. However, using such a perspective could greatly increase the chances of enumerating problems, especially if the system is a black box.

The routes through the attack tree as used in this doctoral research is analogous to such an approach (and could even provide details for future pattern formation), as the techniques used are based on common patterns from known security testing techniques such as flooding. The penetration testing approach used (see Chapter 4) is compatible with studies like that of [52]. The process is performed on a specific target interface (Bluetooth and its implementation), in a specific context (on a vehicle), although the countermeasures are yet to be determined.

2.1.2.4 *Black Box Testing*

Based on the discussions above surrounding the scarcity of technical information, it is clear that a black box approach is necessary. Thus, systematism and structure provides greater coverage (although actual quantification is not possible). Test cases are based on foreseeable abuse cases (since generating test cases from models is not possible in this research). The techniques used are based on known attacks from literature (both vehicular and otherwise) and are analogous to using attack patterns.

Black box testing uses the outside to inside approach and is a general process that is used to probe an opaque system with various inputs [124]. This approach requires only the running system. In an automotive context, there is often no detailed technical information, or information regarding the internal "coherence" of the system [72].

The typical approach has been to concentrate on a specific component or interface, using information that can be acquired through public spaces such as forums or manuals. This approach lends itself to a specialised form of black box testing called penetration testing (see Chapter 4).

2.1.2.5 *Summary of Comparative Methods*

Whilst the coverage of related work seems scattered, they actually represent the facets of testing security. Formal methods are usually the most rigorous. However, as discussed previously, in an emergent field such as automotive cybersecurity, the models or the information necessary to create models to carry out such testing are not readily available. Other methods are less robust, yet they allow us to move closer to enumerating the system and threats to its security. Where tests were performed on real vehicles (such as the demonstrated exploits), the element of realism and proof of a vulnerability's existence could lead to a reconsideration of design elements by those who have access to the resources required for formal modelling. We explore this in Chapter 8.

2.2 STANDARDS AND PROJECTS

With the appearance of experimental analyses on (and attendant appearance of vulnerabilities in) the vehicle, as well as the well-established importance of security testing, the automotive industry has recognised that there is a need to effectively employ both formal and informal testing in a single structured process. Ideally this process should be deployed at every stage in the development, implementation and maintenance lifecycles. As of the time of writing, many of these concepts were incorporated into the seminal "E-safety vehicle intrusion protected applications" (EVITA) project (Section 2.2.1) and the J3061 Cybersecurity Guidebook for Cyber-Physical Systems (Section 2.2.2).

2.2.1 *EVITA*

The "E-safety vehicle intrusion protected applications" (EVITA) project [55] ultimately aims to provide a secure architecture for automotive

on-board networks and evaluates the realisation of this using two "views", the *magnified view* and the *compositional view*.

The magnified view is of especial interest, since within this view, automotive-specific systematic methods of evaluation described. Attack tree modelling (discussed further in Section 4.1) is used to support these processes, although the end goal of verifying whether assets are really protected somewhat differs from the aim of the study which is to identify unprotected assets through a methodical evaluation.

EVITA elaborates on some of the possible usages of the attack tree method under the detailed functional path and mapping approach. Deliverable 2.3 also includes an outline in which security requirements could be traced back to the attack tree, along with what could be gathered from a threat mitigation perspective [133]. This is, broadly, along similar lines to the work done in this project (although our process begins with less knowledge of functionality and other requirements). Additionally, the "dark-side scenario analysis", which is closest to our security testing process, places particular emphasis on risk assessment, whereas the purposes of our own methodology would be to identify specific insecurities relating to an attack goal without looking at the motivations behind it (a necessity for calculating risk).

The compositional view deals with looking at attack categories (and related security guarantees) to ensure that omitted attacks are minimised. The latter is a valuable exercise, however, where a system already exists with unknown properties (and therefore unknown guarantees) as is the case in this study, the ability to analyse coverage in such a way is limited.

Of particular interest is the classification of the severity of various outcomes (Table 1), which could be use in future security assurance cases (see Chapter 6.6).

The classes described in Table 1 denote the potential concerns for stakeholders [55]:

- **Safety** Unauthorised interference with vehicle systems or communications that may impact on the safe operation of the system in question
- **Privacy** Unauthorised acquisition of data relating to vehicle or driver activity, data, vehicle design or implementation

- **Financial** Fraudulent commercial transactions or vehicle theft,
- **Operational** Unauthorised interference with vehicle systems or communications that may impact on operation performance of system in question

There are some seeming mismatches with regards to the translation of severities laterally across the four categories, which might have resulted due to the perspective of manufacturer liability. For example, the classification for severe and life threatening injuries appears comparable to identification of vehicle or driver in the privacy class. Despite this, as part of the seminal standard in the field (J3061), it provides some guidance on framing and contextualising experimental results.

Severity levels have also featured in other projects, such as the "Healing vulnerabilities to enhance software security and safety" (HEAVENS) project [143]. This project aimed to provide threat analysis and risk assessment to facilitate security requirements engineering. It uses the popular threat modelling method STRIDE (a mnemonic for Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege) for threat analysis, ending with the assignment of risk levels based on threat, impact and level of security needed [2]. Its severity levels are similar to EVITA's, in both structure and content, and forms an alternative example of an automotive risk assessment framework [2]. However, there has not been much recent activity as the project completed in 2016 and much of the project information remains confidential.

2.2.2 *J3061*

The seminal (if emerging) standard for cybersecurity in the automotive context is the SAE J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [135]. This standard draws from various concepts known to the automotive industry, such as EVITA and ISO26262 (Functional Safety for Road Vehicles [79]) amongst others. Methodical evaluation methods are also presented in this standard, although information provided has been examples thereof rather than application to a system. Specifically, J3061 also outlines the use of attack trees (in reference to EVITA). The standard also notes that it may only be possible to consider high-level concepts early in the product develop-

Table 1: EVITA Severity Classification for Automotive Security Threats

Severity Classes	*Classes of harm to stakeholders*			
	Safety	*Privacy*	*Financial*	*Operational*
0	No injuries	No unauthorised access to data	No financial loss	No impact on operational performance
1	Light or moderate injuries	Anonymous data only	Low-level financial loss	Operational impact not discernible to driver
2	Severe and life-threatening injuries (survival probable) or light/moderate injuries for multiple vehicles	Identification of vehicle or driver	Moderate financial loss, or low losses for multiple vehicles	Driver aware of performance degradation, or indiscernible operational impacts for multiple vehicles
3	Life threatening (survival uncertain) or fatal injuries, or severe injuries for multiple vehicles	Driver or vehicle tracking, or identification of driver or vehicle for multiple vehicles	Heavy financial loss, or moderate losses for multiple vehicles	Significant impact on operational performance, or noticeable operational impact for multiple vehicles
4	Life threatening or fatal injuries for multiple vehicles	Driver or vehicle tracking for multiple vehicles	Heavy financial losses for multiple vehicles	Significant operational impact for multiple vehicles

ment cycle, a view that has been supported in other studies such as that of Schmittner, Ma, Reyes, *et al.* [139], who demonstrated the application of J3061 to an ECU with remote access capabilities. In light of this, end users such as security analysts or designers could use the systematic evaluation as presented in this thesis as a way of gathering more low-level requirements for the next design iteration.

Work is underway to identify and include integration points in well-established standards such as ISO26262 [74], [139] so that the safety process can be integrated with other processes such as those for security. The converse is also true, in that projects such as J3061

and HEAVENS discuss or refer to parallels in ISO26262. Plans are also underway to update the J3061 standard to further reflect and detail current state-of-the-art surrounding countermeasures to new exploits, the continuing development of the vehicle towards true autonomy and the raft of security testing methodologies and standards (some of which are listed in J3061's Appendix G) from other domains.

3

BLUETOOTH

In this chapter, an overview of the Bluetooth protocol is given (Section 3.1). This is followed by a discussion around known vulnerabilities as well a brief exploration of the state-of-the-art regarding threats to Bluetooth security (Section 3.2). A discussion about how Bluetooth in vehicles differs from conventional computing systems is presented in Section 3.3.

Finally, a threat intelligence study to gain situational awareness surrounding both the automotive Bluetooth interface and attached Bluetooth-enabled aftermarket devices is presented in Section 3.4. This also substantiates the motivation of this research, using Bluetooth as a case study.

3.1 OVERVIEW

Bluetooth is a peer-to-peer wireless communication technology, specified and managed by the Bluetooth Special Interest Group (SIG) [24]. It is more complex than most wireless standards, due in part to the Frequency Hopping Spread Spectrum (FHSS) mechanism designed to reduce narrowband interference. Channel hopping occurs once every 625µs and in some cases also uses Adaptive Frequency Hopping (AFH), whereby channels that can cause interference are avoided [31]. Data whitening is also performed by XOR-ing each packet with a pseudorandom sequence, in order to facilitate signal transmission. For communication to be established between two (or more) devices, nodes are required to synchronise their "hops" and this is done through a process called *pairing* [66] (see Section 3.1.1), for which there are several different mechanisms (such as entering a PIN).

Bluetooth has a complex protocol stack (see Section 3.1.2), with a wide variety of services, although implementations are free to ignore many or all of these. At pairing, an implementation communicates available services through the Service Discovery Protocol profile (see

Section 3.1.4). Each device has a unique address (discussed in detail in Section 3.1.3).

3.1.1 *Pairing*

The pairing process uses one of two mechanisms:

- **Legacy pairing:** The pairing exchange involves the derivation of a link key from the Bluetooth address, the PIN and a random number. This link key is then stored locally and used in all subsequent authentication and encryption processes.

  The PIN is the sole source of entropy for this shared secret [64], and the weakness of this is compounded by the fact that many devices use four digit PINs [120]. However, even with the use of 16-character alphanumeric PINs, there have been many man-in-the-middle (MITM) attacks demonstrated[64], [82], [97]. This has been superseded by Simple Secure Pairing (SSP) in the Bluetooth 2.1 specification, although many old or simple platforms still use the legacy pairing mechanism.

- **Secure Simple Pairing (SSP):** was introduced in the Bluetooth version 2.1+EDR specification [19], and improved on the security of the pairing process by employing Elliptic Curve Diffie-Hellman public-key cryptography [76]. The link key is generated using both public and private keys, nonces and the Bluetooth address of connecting devices.

  Further MITM protection is afforded by either Out-Of-Band (OOB) association, such as using Near Field Communication, or by requiring user interaction [76] as is the case with the Passkey Entry and Numeric Comparison models. Such user-in-the-loop processes are posited to increase security [164]. A fourth association model, called 'Just Works' is usually reserved for pairing with devices that have neither input nor output capabilities, and is the weakest of the four models. However, its importance cannot be overlooked as many aftermarket devices (such as On-Board Diagnostics (OBD) dongles, which afford direct access to the vehicular internal network) may pair in just such a manner.

It should be noted that regardless of the pairing mechanism used, there is no absolute security. Even the use of SSP would not com-

pletely negate the risk of eavesdropping, as an adversary could use a wideband receiver (which monitors all channels simultaneously) in order to gather information. However, there is a significant cost barrier to such devices (upwards of £10,000), and therefore defenders are afforded some breathing space in terms of risk.

3.1.2 *Protocol*

The protocol stack (Figure 1) can be loosely categorised into two classes: the Bluetooth controller (generally the chipset) and the Bluetooth host, with the Host Controller Interface (HCI) acting as an interface to the Bluetooth chipset [162].

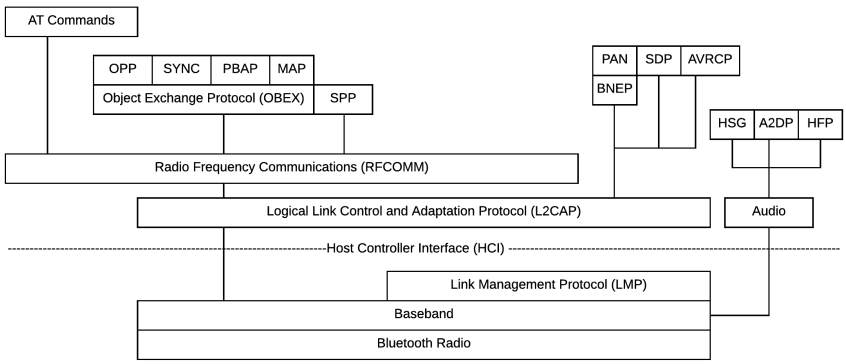

Figure 1: Bluetooth protocol stack (adapted from [142])

The controller deals with the frequency hopping mechanisms, baseband encapsulation and returning the appropriate information to the host, whilst the host is responsible for the protocols in the layers above (Table 2). Descriptions of the profiles sitting on top of RFCOMM and L2CAP are given in Table 5.

3.1.3 *Bluetooth Address*

An essential piece of information in the investigation of this interface is the Bluetooth device ID, also known as the Bluetooth address. This is a 48-bit number used to identify a Bluetooth enabled device, similar in nature to a MAC address. Although theses addresses are supposedly globally unique, counterfeit products (which are not registered

Table 2: Key Bluetooth protocols and their functions (adapted from [142], [162])

Protocol	*Class*	*Description*
Link Manager Protocol (LMP)	Controller	Handles negotiation of low-level encryption, authentication and pairing, and is responsible for regulation of master and slave roles.
Baseband	Controller	Specifies characteristics such as transmission rate and whether the channel is used for transmitting and receiving packets. Generally inaccessible without appropriate bespoke tools.
RF controller	Controller	The Bluetooth radio interface.
Host Controller Interface (HCI)	Link	Acts as a command interface to both baseband controller and link manager, and is generally accessible without custom hardware.
Logical Link Control and Adaptation Protocol (L2CAP)	Host	Used to transport protocols such as RFCOMM to higher level protocols, using a system of channels (where each channel ID or PSM represents a logical endpoint on a Bluetooth device). It can be considered analogous to the User Datagram Protocol (UDP) in that it provides a simple but unreliable, datagram-based transport mechanism, although connection-oriented services are also available [142].
Service Discovery Protocol (SDP)	Host	Identifies and determines services available on devices using a request-response model. SDP runs on top of L2CAP.
Radio Frequency Communications (RFCOMM)	Host	Provides emulation of serial ports, with up to 60 simultaneous connections between two devices possible [142]. Can be used to transport files using the object exchange (OBEX) protocol, send AT commands to phones and supports the IP stack over the Point-to-Point Protocol (PPP).
Object Exchange Protocol (OBEX)	Host	Communications protocol that allows for object exchange between connected devices.

with Bluetooth SIG, or otherwise licensed) could sometimes repeat addresses [68].

The importance of the address lies in the fact that it is used to establish identity, and is important in authentication and synchronisation processes. Furthermore, within a piconet of devices, all slaves transmit using the master's address [31]. Slave to slave communication is not possible; all information is routed through the master device.

As such, the foundation of many of the attacks is predicated on knowledge of this address. If a device is set to *discoverable*, its address is broadcast in the clear. If set to limited discoverability or *hidden*, then the device will respond to a direct inquiry. Otherwise a device is known as *invisible*. However, even when the whole address is not known, scanning even a small range of 100 to 200 addresses could potentially identify devices in the vicinity should some of the address bytes be enumerated [65].

Figure 2: Bluetooth address structure [31]

The address is made up of three parts (Figure 2). The non-significant address part (NAP) comprises the first 16 bits, and as the name suggests, is not involved in any of Bluetooth's functionality. The next 8 bits constitute the upper address part (UAP), used for error checking in packets, and the last 24 bits make up the lower address part (LAP). The NAP and UAP together stand as a manufacturer identifier (also known as the Organisationally Unique Identifier or OUI). The last 24 bits are assigned by the manufacturer, ensuring that they are (supposedly) unique within the device set.

The NAP and UAP information could be used to identify the manufacturer of either the device or the Bluetooth chipset integrated into the device (a list is publicly available on IEEE's Standards Register [77]). This is a precursor technique to determining manufacturer specific issues or flaws that could be used in exploitation planning. Conversely, because brute-forcing to determine the entire 48 bit address is not feasible, a manufacturer's identity could be used to narrow down the field of possible addresses [65] using a tool such as *RedFang* [156].

3.1.4 *Profiles*

Sitting on top of various base protocols are Bluetooth profiles, which generally describe end user applications and their general behaviours. Profiles consist of information regarding dependencies, user interface formats and specifically required stack protocols to be used by the profile [20]. Bluetooth standards specify various service profiles that could be used in order to customise the technology, whether that be to enable "hands-free" communication, allow file transfers or grant access to phonebooks and messages [20]. This information is vital in detailing what the device is capable of doing, and, from an adversary's point of view, also gives information on potential weaknesses.

Profiles can be standard (as specified by Bluetooth SIG), or bespoke according to the needs of the manufacturer. Discovery of these profiles is usually performed through the use of the Service Discovery Protocol (SDP). A list of profiles typically found in vehicles is given in Section 3.3. A list of currently adopted and supported profiles can be found online [20].

3.2 VULNERABILITIES

There are many attacks that could be performed. These attacks can be classified into several categories including surveillance, data access, man-in-the-middle (MITM) attacks, denial of service (DoS), obfuscation, fuzzing and sniffing (Table 3).

As an entry point, Bluetooth has much potential as a viable target: the initialising stages are particularly vulnerable, due to the use of "discoverable" modes (where devices broadcast their existence), the lack of hopping during these stages and potentially weak encryption during the pairing process [64]. These are all well researched, however, the focus has been predominantly on the technology itself, rather than how it applies with regards to the implemented system in the vehicle. Attacks such as brute-forcing the PIN has already been demonstrated (in experimental settings) to lead to privacy attacks or attacks on in-vehicle networks [36].

There are fundamental assumptions in many implementations of Bluetooth that can be exploited as part of any testing process. For example, the assumption that the short range of Bluetooth provides

a measure of security, or that once a connection is established (i.e. as a trusted device), the connection remains permanently secure.

The attacks described in Table 3 echoes the trends apparent whilst surveying the Common Vulnerabilities and Exposures (CVE) database (Table 4), as well as the literature surrounding Bluetooth vulnerabilities.

Many of the attacks are older (dating back to 2004), with the vast majority targeting the pairing system, including brute-forcing the PIN [36] or sending inaccurate information about the capabilities of each device to force a 'Just Works' association model (the No Input No Output or NINO attack), which has, relatively, very little security [76].

How devices store information is also targeted by tools such as *nOBEX* [113], which performs fuzzing through Bluetooth's synchronisation process, or by using long device (HCI) names to cause detrimental effects. The more severe attacks focused on the Object Exchange (OBEX) protocol, especially with the presence of the File Transfer Profile (FTP) [111], as this allows binary object exchange. A few used crafted applications and packets, such as transmitting a Trojan through a paired Android phone [36]. There has also been a vulnerability recorded on gaining access to a Bluetooth enabled device by misrepresenting the user interface and allowing an attacker specified message, thereby tricking a user into granting the necessary permissions (CVE-2006-1367). This is also known as a *Blueline* attack and is a particularly interesting point when considering the fact that the automotive front-end does not give a great deal of feedback even when actions are being performed on it. More information on Bluetooth security testing tools and methods is presented in Chapter 5.

Table 4: Relevant recorded CVE Bluetooth vulnerabilities [145]

CVE ID	*Platform*	*Vulnerability and method*
2004-0143	Nokia 6310	Causes DoS via sending of malformed OBEX messages.
2005-0681	Nokia Symbian 60	Cause DoS by configuring device with long Host Controller Interface (HCI) name.
2005-1333	Before Mac OSX 10.3.9	Directory traversal by exploiting undisclosed OBEX vulnerability.
2005-2250	Nokia Affix 2.1.2	Execution of code via buffer overflow caused by long HCI name.

Table 4: Relevant recorded CVE Bluetooth vulnerabilities [145]

CVE ID	*Platform*	*Vulnerability and method*
2005-2547	Bluez 2.16-2.18	Use of shell metacharacters in device HCI name.
2005-3093	Nokia 7610, 3210	Cause DoS via use of certain characters in filename when transferring files via OBEX.
2006-0670	Bluez hcidump 1.29	Cause DoS via sending of malformed L2CAP packets.
2006-1367	Motorol PEBL U6	ASCII Terminal (AT) level access for data extraction by tricking user into granting permissions via user interface misrepresentation.
2006-6895	Sony Ericsson T60	Obtain unauthorised enquiry responses via exploitation of improperly implemented "limited discoverable" mode.
2006-6898	Before Widcomm Bluetooth for Windows 4.0.1.1500	Allows listening and recording of all information, via passkey guessing based on manufacturer. Requires static PIN in implementation. Also known as "CarWhisperer".
2006-6908	Widcomm 3.x	Cause DoS or execution of code via buffer overflow in Bluetooth stack COM server.
2007-0521	Sony Ericsson K700i, W810i	Cause DoS via repeated pushing of file via OBEX push.
2007-0522	Motorola MOTORAZR V3	Cause DoS via repeated pushing of file via OBEX push.
2007-0523	Nokia N70	Cause DoS via repeated pushing of file via OBEX push.
2007-0524	LG Chocolate KG800	Cause DoS via repeated pushing of file via OBEX push.
2007-3753	iPhone 1.1.1	Cause DoS or execution of code via crafted Service Discovery Protocol (SDP) packets.
2008-4295	Windows Mobile 6	Cause DoS by configuring device with long HCI name.
2009-0244	Windows Mobile 6 Pro	Directory traversal using ../ [dot dot slash] in filename when using OBEX file transfer protocol.
2011-4276	Android 2.3-2.3.6	Obtain contact data via AT phonebook transfer, also known as "Bluesnarfer".

Table 4: Relevant recorded CVE Bluetooth vulnerabilities [145]

CVE ID	*Platform*	*Vulnerability and method*
2014-4354	Before Apple iOS 8	Undisclosed exploitation if Bluetooth is enabled when upgrading platform.
2014-4428	Before Mac OS X 10.10	Spoof previously paired device through lack of requirement for encryption in low energy human-interface devices.
2014-7914	Before Android 4.4.0	Issue HCI commands without pairing via crafted OOB handover.
2015-1106	Before Apple iOS 8.3	Data extraction through discovery of passcodes by reading the lockscreen.
2015-3683	Mac OS X 10.10.4	Execution of code via crafted application.
2015-3847	Before Android 5.1.1 LMYA8T	Allows removal of stored SMS messages via crafted application.
2015-6613	Before Android 5.1.1 LMYA8X and 6.0	Allows sending of commands to a debugging port to gain privileges via crafted application.
2017-0423	Between Android 5.0.2 and 7.1.1	Allows remote privilege escalation through a separate exploit on the Android Bluetooth stack.

3.3 BLUETOOTH IN VEHICLES

The wireless nature of Bluetooth has been attractive to automotive manufacturers as a way of reducing weight and wiring in the vehicle, along with the hands-free services that Bluetooth can offer. The latter is driven in large part by the advent of regulations barring the use of mobile phones in vehicles in the United Kingdom. Its flexibility means that manufacturers can offer customised features to end users. These advantages mean that Bluetooth is now a ubiquitous technology in vehicles and is deployed in millions of cars.

A summary of the standard profiles typically included in vehicles is summarised in Table 5, and where they sit on the Bluetooth protocol stack can be seen in Figure 1. The list given here is by no means exhaustive, but is an indication as to the standard features implemented in automotive headunits. Note that even though the profiles

Table 3: Bluetooth attack classification (adapted from [44])

Attack classification	*Threats*
Surveillance	Includes general scans (or war-nibbling), inquiry scans and brute scans to determine non-discoverable addresses. Manufacturers can be profiled using organisationally unique address bits. Also includes service enumeration.
Range extension	Most consumer devices (including automotive implementations) are Class 2, with a range of up to 10 metres. Range can be extended through the use of external directional antenna or passive radio locators.
Obfuscation	Includes spoofing or cloning a device name, class, address or service profile fingerprint. Can serve to further other actions such as man-in-the-middle attacks.
Fuzzing	Injection of arbitrary or malformed data.
Sniffing	Using Bluetooth narrowband or wideband receivers or tools in order to dump raw data from a connected Bluetooth interface.
Denial of service (DoS)	Flooding with data, or jamming signals to cause applications or devices to freeze or crash or battery exhaustion.
Malware	Infection from malicious programs via Bluetooth interface.
Unauthorised direct data access	Includes targeting hard-coded default PINs, brute forcing PINs, targeting vulnerable implementations of APIs, sending commands via covert channels to extract data, or using loopholes in the object exchange (OBEX) protocols.
Man in the middle (MITM)	Masquerading as a trustworthy entity, or injecting oneself in the middle of a communication in order to eavesdrop on or modify data, as described by [64].

themselves may still be in use, profile *versions* could still be deprecated, especially if vehicles use legacy Bluetooth specifications.

Bluetooth implementation on vehicles differs from conventional computing and mobile platforms. The software present on vehicles may not have been updated in years and older chips are in use even in newer vehicles, with many still using legacy pairing, or having backwards compatibility to legacy pairing. Presented information is customised by device and not by users (so no distinction is made between users of the same remote device) with user information potentially centrally held on the infotainment unit [127].

Although it has been posited that requiring user interaction within the authentication process increases security [164] (for example with numeric comparison), many vehicles use other pairing mechanisms such as passkey entry (some with a default universal static PIN [44], [119]). The front-end of the system may not ask for user confirmation or display alerts as might be expected in other embedded systems, such as mobile phones. This is likely due to the node containing the screen or other outputs being separate from the ECU containing the actual operating system. An example of this is where the pairing of a device (that is not categorised as a smartphone) is sometimes absent from the "paired devices" list on screen, even where there is an active connection.

Additionally, a vehicle is mobile and is rarely stationary with the ignition or engine turned on. This, combined with a relatively short range of ten meters could pose a challenge to an attacker. However, range extension (see Table 3 in Section 3.2) has been used successfully to extend the range to about a mile, which led to the testers being able to inject audio and eavesdrop on in-cabin conversations [167]. Furthermore, compromise could also occur pre-travel (for example in a car park or a garage) for possible disruption later on. With premeditation and preparation, an attack could also involve following the target vehicle so that it stays within range.

The majority of built-in infotainment systems either search for a device to pair with or require a user to actively enable Bluetooth [120], though the seeming security of the latter is diminished given that not every vehicle limits the time in which the interface is discoverable. Additionally, many implementations look for previously paired devices and may initiate a connection without switching on the discoverable mode; potential attackers could also test for the existence of

Table 5: Typical Bluetooth profiles in vehicles. All information comes from Bluetooth SIG [20] and the Open Mobile Alliance [144]

	Profile	*Abbrv.*	*Function*
Hands-Free and Audio	Hands-Free Profile	HFP	Provides voice connections and remote control of the connecting device in conjunction with a hands-free device.
	Headset Profile	HSP	Provides support for audio between two devices (headset and mobile phones). Has limited AT command support.
	Audio/Video Remote Control Profile	AVRCP	Provides remote control functionality to any accessible audio or video equipment.
	Advanced Audio Distribution Profile	A2DP	Provides procedures for distribution of high quality audio (which is distinguished in the specification from voice audio).
Data exchange / synchronisation	Serial Port Profile	SPP	Provides the ability to create a serial connection by emulating an RS-232 connection.
	File Transfer Profile	FTP	Provides functionality for transferring files between devices.
	Phonebook Access Profile	PBAP	Provides the ability to exchange Phone Book objects (such as contacts).
	SyncML	-	Former name of an open information synchronisation standard, currently managed by the Open Mobile Alliance. The latest specification is dated 2007 (OMA SyncML v1.2.2), and no further specifications have been announced or released. Provides information synchronisation (such as calendars or contacts) between mobile devices.
	Message Access Profile	MAP	Provides ability to exchange messages between devices.

a device via a name inquiry, to which a device in limited discoverable mode will respond. An adversary could then wait for the opportune moment once the existence of a device is known to pair and form a connection with the target.

3.4 THREAT INTELLIGENCE

This section presents a threat intelligence study with regards to the availability and characteristics of Bluetooth devices in the automotive context in real-world driving conditions.

The characterisation of Bluetooth devices within a certain range has been studied [26], with data gathered in order to perform statistical analysis on the location and frequency of types of Bluetooth devices that might be found whilst driving. However, their primary focus has been to concentrate on behaviours that could be used to build up to a vehicle-to-vehicle or vehicle-to-infrastructure scenario. Whilst useful for general awareness, the approach used here is based on a security perspective.

Most work on Bluetooth security is on weaknesses inherent within the Bluetooth standard itself, whether that be:

- Looking at vulnerabilities during the capability exchange phase, where connecting devices inform each other as to their input and output capabilities (i.e. the NINO attack) [76];
- On certain association models such as passkey entry [9] or 'Just Works' [64];
- On aspects of the underlying communications protocol [32], [123].

However, practical implementation could also introduce security problems to the underlying (operating) system, especially if it includes the use of legacy or deprecated specifications as is the case in this research (see Chapter 7).

Experimental analysis of the automotive implementation of Bluetooth has been explored by a few studies [36], [161], although it has been limited to a single version or implementation (see Section 2.1.1). Similarly, work has been done on enumerating specific vulnerabilities in a specific on-board diagnostic device [7], but do not concentrate on where or when such a device might be encountered in normal driving conditions.

The closest work in gathering information security-specific information from a large number of vehicular implementations is that of Oka, Furue, Langenhop, *et al.* [120], who performed a survey on the state of discoverability and the types of Bluetooth pairing in both vehicles and aftermarket devices by examining publicly available manuals. They found that a significant proportion of vehicles still used four-digit PINs (which were in some cases unchangeable) and that the vast majority of aftermarket devices used legacy pairing. This is extremely useful information, especially for situational awareness, and the endeavour here is to build on this work by examining real vehicles in real driving conditions.

3.4.1 *Data Collection*

Two surveys were used to investigate the versions of Bluetooth on a vehicle, one, in-cabin inspection, required the knowledge of the vehicle owner, and the other (war-nibbling) gathered information at a broader (but shallower) level.

The easiest way to identify the pairing mechanism in use is through the use of a sniffer. Even a low-cost Bluetooth transceiver can be used to identify such devices, as the Bluetooth address and some of the information about the device is publicly broadcast (if the device is 'discoverable') and available for inspection.

3.4.1.1 *In-cabin inspection*

To find the Bluetooth version (and to a certain extent the chip manufacturers) on a vehicle at a granular level, it was necessary for Bluetooth to be enabled and for the infotainment unit to be discoverable and within range. This necessitated being inside the vehicle cabin. With regards to generalisation, although each make and model may have a different Bluetooth implementation, there are sufficient numbers of each type on the road to make the results meaningful.

The criteria for vehicle selection were the vehicle age (2010 or newer, to ensure on-going relevance), and that it be road legal (i.e. not test vehicles). Only Bluetooth-enabled infotainment systems were considered as we were interested in the default (Bluetooth) security status of the vehicle. The inspection used the *hcitool* suite [94] to request information from the headunit.

3.4.1.2 *War-nibbling*

This method sets a tool on monitor mode to see all relevant devices and is considered to be a subset of wardriving [44]. War-nibbling was used as a general survey mechanism to find the number of vehicles that:

- Allowed for discovery on the road in real driving conditions;
- Used legacy pairing and
- Were visible for a duration (which we have categorised as less or more than a minute).

The third observation is especially significant from a threat intelligence point of view, as the longer the duration, the more likely that an attacker is able to connect and perform malicious actions. Anything over a minute is enough to initiate a pairing, perform a port scan, or perform flooding-type attacks, assuming some level of premeditation and preparation on the part of the adversary (demonstrated in Chapter 7).

Monitoring was repeated over 28 trips, ranging from 20 to 60 minutes long and encompassing mostly urban areas. This included two carparks, two different highways and three different town centres. Note that the number of vehicles or devices surveyed in any given area could be increased by using range extension antennas [44], with distances of over a mile reportedly possible [64]. However, this was not feasible due to legal considerations.

Identifying information regarding a Bluetooth-enabled device that has been set to discoverable mode includes the:

- **Bluetooth address**: which can be used to identify the Organisationally Unique Identifier (OUI). Recall that this is the first 24 bits of a Bluetooth address that can be used to identify a registered manufacturer. The register of OUIs can be accessed publicly through IEEE's Registration Authority directory [77].
- **Bluetooth alias**: also known as the Bluetooth name, which is customisable by either manufacturer or user (if implementation allows).
- **Bluetooth class of device**: takes the form of six hexadecimal digits. These define the general functionality class for a device.

There are 32 major classes (e.g. networking, rendering, telephony) with any number of minor classes in the context of the major class assigned to a device. Whilst this gives generally useful information regarding intended device usage, a device may contain functionality beyond the advertised class.

- **Bluetooth service profiles**: includes specific capabilities that the device is set to deliver. Recall that these profiles are specified by Bluetooth SIG and are standard to anything that implements the Bluetooth protocol. Note that manufacturers can choose the combination of profiles to implement, as well as implement bespoke profiles.

As the scan picked up all Bluetooth devices in the vicinity (including smartphones and laptops), devices deemed to be within scope for this study were filtered based on the acquired Bluetooth information fulfilling at least two of the following criteria for vehicles:

- If the Bluetooth alias contained the name of an automotive manufacturer, vehicle model or licence plate number;
- If the Bluetooth class indicated that it was a hands-free device with telephony, rendering, object transfer or audio capabilities (generally device class 0x240408 or 0x340408);
- If the OUI indicated that the manufacturer is a known supplier of automotive headunits.

Aftermarket devices were identified based on Bluetooth information fulfilling at least one of the following criteria:

- If the Bluetooth alias contained the name of a known aftermarket device (e.g. GPS, Radio, OBDII) or the name of an aftermarket carkit;
- If the Bluetooth class indicated that it was capable of audio, rendering or networking (generally device class of 0x200408, 0x240408, 0x280408, 0x340408 or 0x420300). This is the loosest of the three criteria, as even something that indicates a GPS unit could have a class of 'uncategorised';
- If the OUI indicated that the manufacturer is a known supplier of aftermarket devices.

This method allowed for greater numbers of vehicles to be surveyed. The scan is based on area and range. Because close examination of the vehicle is not possible with this form of survey, the exact age, make and model of the vehicles in question cannot always be determined. Some of this information, however, can be inferred by looking at the OUI of the Bluetooth address that is broadcast, the device class and the device name (which may have manufacturer names in the same way that broadcasting from a phone might lead to a phone model being made known). Nevertheless, it is a useful starting point in a real world situation, allowing for an approximate measure of the proportion of vehicles on the road using the more insecure form of legacy pairing.

Both the inspection survey and the war-nibbling were carried out using a standard *Kali Linux* distribution [118], using the *BlueZ 5.x* Bluetooth stack [15], with a Cambridge Silicon Radio Bluetooth 4.0 Class 2 transceiver.

3.4.2 *Data Analysis*

Presented below is an analysis of results from the two methods to carry out threat intelligence as described above.

3.4.2.1 *Inspection*

A large variety of Bluetooth versions were seen in vehicles. As can be seen in Table 6, there is a general trend towards the newer versions of Bluetooth as the age of the vehicle decreases, although the insecure version 2.0 (deprecated in 2014) was found in a 2016 registration.

Furthermore, we observed that some vehicles with Bluetooth version 2.1 or higher offered backwards compatibility to legacy pairing, complete with the use of a default four-digit PIN, although we did not investigate this systematically[1]. This could allow an adversary an easy connection to the target vehicle, especially if vehicles used easily guessable PINs such as "0000" or "1234" [120]. Furthermore, there is a significant lag between the year of vehicle registration and the year that each version of Bluetooth was adopted (see Section 3.4.3.1).

There was no correlation between the price of the vehicle (as a measure of whether it was a "premium" vehicle) and the version of

1 Further investigation would have required attempting to open active connections to the vehicle, and not all vehicle owners gave this permission.

Table 6: Survey of Bluetooth versions implemented in automotive infotainment systems

Bluetooth in the vehicle. There was also no pattern between chipset or Tier 1 supplier and the versions of Bluetooth they provided.

We examined 11 different makes of vehicles, which corresponded to 14 different headunit and six different Bluetooth chip providers. This showed a wide variety of combinations of the three provider types (vehicle, headunit and chipset). Furthermore, even though the survey sample here was small, there are similar indicative trends even within the same manufacturer (Table 7). Note also that each single vehicle of a manufacturer and model observed represents a large number of vehicles on the road, which we reasonably assume would have the same characteristics since production lines are standardised. However, we make no claims of statistical significance, but rather point out interesting indications of possible trends.

3.4.2.2 *War-nibbling*

All devices found using this method were publicly broadcasting their Bluetooth addresses. Hidden devices would require an active probe

Table 7: Bluetooth versions across registration years

to a predetermined Bluetooth address. Generally, the duration in proximity is too short for such an enquiry method when driving. A summary of the results can be seen in Table 8.

122 vehicles were found that broadcast their address within the 28 trips monitored. Like the inspection survey, one of the main points of analysis in the war-nibbling exercise was around the number of devices that used legacy pairing (thereby implying the use of Bluetooth 2.0 or older) as a proportion of discoverable devices.

The majority of vehicles found (53.2%) used legacy pairing exclusively, which has been shown to be vulnerable to MITM attacks (see Section 3.2). This was common across certain OUIs, which correspond to potential Tier 1 suppliers. These may be indicative of the age of the vehicle, especially if a large Original Equipment Manufacturer (OEM) has used a different Tier 1 for each particular age range of vehicle model. Similarly, the majority of aftermarket devices found (55.1% of the 98) also used legacy pairing only and likewise had certain OUIs in common.

Of the 122 vehicles found, 31 (25.4%) broadcast the name of the device, which in all cases was also the make or model of the vehicle. Two of these device aliases also contained personal names. One vehicle broadcast what seemed to be a PIN number of '1234'. A small number (seven) also broadcast the services on offer through Bluetooth, with the most common being the Hands Free Profile (HFP), Serial Port Profiles (SPP), Object Push Profile (OPP) and Phonebook

Table 8: Survey of automotive Bluetooth devices through war-nibbling

Access Profile (PBAP). The presence of SPP could also allow for commands to be sent through via 'attention modem' (AT) commands (see Section 7.4.2); this is also known as 'Bluebugging' [4]. The presence of OPP has been shown to lead to Bluesnarfing (connecting to these ports and spamming with data) [111].

Of the 98 aftermarket devices, 31 (31.6%) broadcast the name of the device, which in all cases indicated the nature of the device. Three of these devices broadcast license plate numbers, which enabled us to find information such as tax status, engine size and even pictures of the vehicle in question using a search engine. Three were named 'OBDII', which indicated a device with a direct connection to the internal vehicular network. A further seven out of these 31 devices (all navigation units) also broadcast the device serial number. There was only one OUI in common between vehicular units and aftermarket devices. Both vehicle OUIs and aftermarket OUIs varied greatly with 17 and 14 different manufacturers found respectively.

Bluetooth-enabled vehicles and aftermarket devices were more likely to be found in town centres. This is where traffic is moving slowly enough such that the short range is less of an issue. Many of the addresses found in car parks were that of aftermarket devices, which

could stay powered (if, for example, it is connected to an OBD port) even when the vehicle itself is turned off.

Duration of visibility speaks more to the nature of the location where data gathering took place. We would envision that an attacker serious about compromising a vehicle would either follow the vehicle within the necessary range, or use range extension methods. However, it is useful to see whether the possibility exists and in what (general) proportion compared to discoverable numbers. We found here a small portion where visibility was long enough to at least initiate the pairing sequence. As expected, they were generally in places where the traffic was either slow-moving or static due to the short-range of the Bluetooth scanning used.

The total number of devices found was only a small sample with regards to the actual number of cars within the scan range; this could be due to the use of relatively low-powered equipment. There was also a more substantial prevalence of certain manufacturers (whether it be built-in or aftermarket systems), although this could reflect the popularity of those makes and models. Additionally, the vehicles found through scanning could be set to broadcast by default by certain manufacturers. If that is the case, then that would skew what was found towards those manufacturers. However, statistical analysis between sales rates of makes and models and its relation to the vehicles found is currently out of scope.

3.4.3 *Discussion*

This discussion encompasses two themes: Bluetooth implementation in vehicles (Section 3.4.3.1) and the aftermarket devices that were recorded during war-nibbling sessions (Section 3.4.3.2).

3.4.3.1 *Bluetooth in vehicles*

Although the trend is towards more secure forms of Bluetooth as vehicle age decreases, even the oldest vehicles surveyed in this study still have many years left on the road, with the average lifetime of a vehicle ranging from ten to fifteen years [88]. This means that the increased security risk from having implemented legacy pairing will be present for some time yet. It should be noted that, although SSP improves on the security of the pairing process, it is by no means completely secure. There are demonstrated attacks on many aspects

of both the pairing mechanism and the protocol itself (as discussed in Section 2).

There is also the wider discussion of just how far technology implemented in vehicles trails behind other contemporary embedded systems. Bluetooth version 1.2 (found in one 2010 vehicle) was adopted by the Bluetooth Special Interest Group (SIG) in 2003 [24] and withdrawn in 2009 [22]. Bluetooth version 2.0 (the last version to implement legacy pairing), found in a vehicle registered in 2016, was adopted in 2004 [24] and deprecated in 2014 [23]. Bluetooth 2.1 was adopted in 2007 [23] with the headline feature being the switch from legacy pairing to secure simple pairing. Bluetooth versions 4.2 and 5, the most recent adoptions (in 2014 and end of 2016 respectively [24]) were found in none of the vehicles.

This technology lag could be due to many reasons. The long lifetime of vehicles [88] and the difficulty of updating vehicles [115] would mean that by the time the vehicle left the road, the software may be years old. Long vehicle design cycles [125] also contribute to this lag, as adoption of specifications might not be realised in implementation until years later, by which point versions of the technology adopted may have been updated, patched, deprecated or withdrawn as is the case here. Furthermore, reuse across models and ages is a common cost-saving approach [125] and could mean that legacy software persists even in the newest vehicles.

3.4.3.2 *Aftermarket devices*

There was a plethora of different devices (see Table 9). Not all devices are necessarily directly connected to the vehicle, however, any tampering would still interfere with the driving experience.

Many of the aftermarket devices also broadcast the nature of the device. Where a device name is available, a quick search on the Internet reveals manuals and other such public documentation that could be used to gain more information. For example, we found a particular car kit brand (nine examples of which were found) where SSP was in use by default, but which nevertheless stated openly in their manual that '`0000`' as a PIN could be used should the connecting device use legacy pairing only. Also of interest were those with legacy pairing and named devices that have been demonstrated to affect the vehicle, such as "OBD II" [36] or similar. Such devices are the most dangerous. Assuming this device stays within this range, pairing could be

Table 9: Types of Aftermarket Devices Found

Some materials have been removed from this thesis due to Third Party Copyright. The unabridged version of the thesis can be viewed at the Lanchester Library, Coventry University.

initiated which could lead to access to the car's internal network. In these cases, access to the official CAN database used by that vehicle is unnecessary. This is because there are public diagnostic test messages or CAN messages that could be reverse engineered through trial and error that could be used in a dangerous manner (see Chapter 7). Even if no attacks were used, there may be privacy concerns, as the vehicle could be queried wirelessly for details such as its Vehicle Identification Number (VIN), the diagnostic command for which is standard (see Chapter 7.4.4)).

Aftermarket devices present a problem in that manufacturers have no control over their design or deployment, yet they extend the system boundary of the vehicle another step outward. This is especially dangerous if the connection is wireless, since an in-cabin presence is not needed, thereby removing a physical barrier from an attacker. Aftermarket devices therefore make a car more insecure for the following reasons:

- Firstly, if the vehicle had no wireless interfaces, these devices open up a vehicle to possible wireless attack;
- Secondly, even if the vehicle was relatively secure on all fronts, the introduction of an aftermarket Bluetooth-enabled tool with dubious security could compromise the security of the car as a whole, especially if the aftermarket device has a direct connection to the in-vehicle network; and
- Finally, the plethora of devices means that there is a threat to privacy, as footprinting of individuals becomes easier (for ex-

ample, if these devices broadcast license plates, serial numbers, or even the unique Bluetooth address of the device itself). If the device is a diagnostic or OBD-II connected device, then other information of value could be extracted, such as the VIN (Vehicle Identification Number).

3.4.4 *Conclusion*

A large proportion of vehicles surveyed through inspection and warnibbling used legacy pairing exclusively. This was true across a range of manufacturers, models and ages. The implication is that these vehicles are more at risk of a pairing compromise than through the use of SSP. Similarly the majority of aftermarket devices found were also only capable of legacy pairing, and this represents an increased risk as aftermarket devices can make a vehicle more insecure. Furthermore, even if SSP was the default pairing mechanism, publicly available information of such systems (through manuals) openly state the use of default easily guessable PINs if the connecting device did not have SSP capabilities. This means that these devices are susceptible to a downgrade attack, and the risk of compromise is again increased. Finally, the study here demonstrates a very real risk to vehicles in the real world through Bluetooth interfaces, and therefore motivates and justifies the research presented in this thesis.

Part II

SYSTEMATIC EVALUATION

"A computer once beat me at chess, but it was no match for me at kick boxing."

Emo Philips

4

METHODOLOGY

The methods in use in this thesis (attack trees and penetration testing) are used to evaluate a vehicle through the use of a software tool. This tool (which is detailed in Chapter 5) is designed to assist a security tester by automating some parts of the systematic testing process. In this chapter, an introduction to both attack trees and penetration testing is given, focusing on the parts we make use of in later chapters.

The general definition of evaluation is the formation of a judgement. This judgement is dependent on the nature of the evidence collected and the assurance that can be derived from this. More specifically, in this thesis, evidence gathering and the classification or systematisation thereof (the eponymous thesis title) forms part of the security assurance case. Ultimately, the aim is to provide guidance as to what and where the problem might be in the system under test.

The methods used in this doctoral research are empirical, and derived from standard practice in the security industry, namely threat modelling and analysis using attack trees, with penetration testing techniques employed to populate these trees. The premise of these methods is founded on testing a system from an attacker's point of view to identify system weaknesses and to reflect what an adversary might face in reality. Attack trees (the foundations of which are described in Section 4.1) were used to provide systematism and traceability. Furthermore, the use of attack trees has been presented by the cybersecurity standard J3061 (Chapter 2) as one of the more useful techniques for security testing in the automotive context.

Because there is no technical information available for this research regarding the systems under test, the process mapped out by these attack trees is a specialised form of black box testing called penetration testing, the concepts of which are further explored in Section 4.2.

4.1 THREAT MODELLING

Automotive security is a diverse field, with full functional specifications unlikely to be readily available due to commercial sensitivity.

Combined with the fact that there is little work to build on (see Chapter 2), the lack of information necessitates a black box approach.

Threat modelling is the process by which security threats can be determined, analysed and documented [103]. Many threat models can broadly be taken to represent the decision making process of a potential adversary. A threat can be defined as any potential harmful event that could compromise an asset (an object of value) [51]. These assets can be physical, digital or human. Combinations of attack vectors and methods are usually employed in order to realise these threats.

This process typically follows the process of identifying a threat (synonymous in this case with an attacker goal), which can be broken down into sub-goals iteratively until individual actions are identified [103]. Popular methods include Microsoft's STRIDE (a mnemonic for the threat categories of spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege), DREAD (damage potential, reproducibility, exploitability, affected users and discoverability) and visualisation tools such as Data Flow Diagrams (DFDs) [99], [103] all of which help to classify, assess the risk of and visualise the threat landscape. These methods can be used in combination with constructions such as attack trees in order to further enumerate the threat, such as in the work done by Klöti, Kotronis, and Smith [89]. The attack tree used in the paper (which might be instead subsumed under a different category of vulnerability tree [92]) explored exact paths to the pre-discovered vulnerability [89]. One crucial difference here to such work, however, is that vulnerabilities had already been identified in an emulated environment where the full system was known, which is not the case here.

There are also structures other than attack trees that model security-related testing processes. Examples include attack nets, which are customised Petri nets with places representing states or modes of interest, and transitions that represent events such as input or commands [109]. Although eminently suited to singular activities, such as bringing together seemingly unconnected flaws to form an individual attack path, representing relationships between different attacks (especially on poorly documented systems) is more challenging [109].

4.1.1 *Definitions*

Flaws are hard to define. Although a flaw can be described as any "demonstrated undocumented capability, which can be exploited to violate some aspect of the security policy" [155], this doesn't strictly capture all the nuances of exploitable vulnerabilities, including misconfiguration, mismanagement or mishandling of error messages (resulting in information leakage), or techniques such as spoofing, which may not depend on the vulnerability of the spoofed system, but on the peculiarities of the victim of the attack (which may be human rather than a machine).

More accurate would be Bishop's [13] interpretation of an insecure system: any method, weakness, vulnerability or tool that forces a system to enter a disallowed state, or results in a successful execution of a disallowed action, as defined by a security policy, can be defined as insecure. There are still ambiguities with the word "disallow", since this implies an "allower" who has knowledge of such states. The perspective taken in this thesis is instead a slight variation, that an insecure system is one where it can be forced into either a disallowed *or* unintended state.

Note that the words *flaw*, *vulnerability* and *weakness* are often used interchangeably in many definitions. Since there is no consensus for definitions of these words, for the purpose of this thesis, we use the following:

- A weakness is an element or lack thereof in a system which increases the risk of exploitation. This implies that not all weaknesses are exploitable.
- A weakness can be caused by an error in implementation or could be inherent in primitive designs. Thus a security flaw is a subset of a weakness.
- A vulnerability is a specific instance of that weakness (i.e. a definite exploitation of weakness).

Pulling the above together, an example of a weakness in a system would be the ability to write to a buffer with no boundaries. However, illustrating the point that a weakness is not always exploitable, Checkoway, McCoy, Kantor, *et al.* [36] found an example of where a

buffer overflow would not work in practice (with the aqLink telematics protocol) due to insufficient bandwidth.

Where exploitation *is* feasible, the presence of the ability to write to a buffer with no boundaries (the weakness) then leads to the vulnerability: a specific buffer that can be overrun by sending in specific content to achieve specific actions.

These terms and the example thereof are analogous to definitions by Bishop and Bailey [14], which characterises a generic vulnerability (i.e. a weakness) as a set of vulnerable states, with a specific vulnerability characterising a specific vulnerable state. This is also reflected in how databases such as the Common Weaknesses Enumeration [146] and Common Vulnerabilities Enumeration [145] databases are set up, with the former describing classes or categories of vulnerabilities, and the latter describing specific vulnerabilities on specific implementations.

4.1.2 *Attack Trees*

Attack trees were first developed to describe the security of systems [140] in a structured manner and are conceptual diagrams meant to illustrate threats from an attacker's point of view. Early versions of the attack tree looked at taxonomies of attacks. Since then expressive models have been developed in order to infer explicit or implicit links, and develop causality. This makes it possible to simulate multi-stage or co-ordinated attacks [39] , build multi-parameter attack trees [28], adapt to executable test cases [104] and take into account imprecise data [85]. However, many of these techniques have in common the inability for full automation due to the human element involved in many attacks.

These trees can be represented diagrammatically (Figure 3) or textually (Figure 4). Attack trees focus on abuse cases (in this case an attack), and even in an informal capacity, can support threat assessment. This information would usually need to be further formalised (see Chapter 8), empiricised or investigated (if resources and available data permits), but is nevertheless a useful starting point for threat identification [121].

Enhancements are available for attack trees, a useful one conceptually being the Stratified Node Topology as proposed by [39], in which parts of the hierarchy of the tree are assigned levels. The leaves are

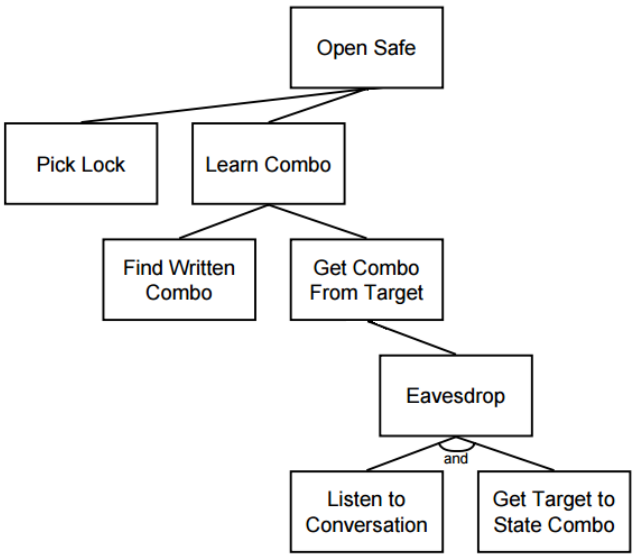

Figure 3: An example attack tree detailing how to open a safe [140] in [33]

```
Open Safe
├── OR: Learn combo
│   ├── OR: Find written combo
│   └── OR: Get combo from target
│       └── OR: Eavesdrop
│           ├── AND: Get target to state combo
│           └── AND: Listen to conversation
└── OR: Pick lock
```

Figure 4: Textual representation of an attack tree detailing how to open a safe [140] in [33]. Notation has been used in other studies [30].

referred to as event levels (and are comparable to the attack methods in our tree), the intermediate steps are called state levels (comparable to attack patterns) with the root being the top level (synonymous with attacker goal). Attack trees can also be considered analogous to the more common concept (in automotive engineering) of fault trees. The primary difference between the two structures can largely be attributed to paradigm. Where a fault tree looks at random faults that could cause an undesired event, the attack tree concentrates on intentional malicious actions that could cause the system to enter an undesired state [27].

The attack tree can be extended until all contextual requirements are met [136]. This process begins with elucidating on the number of conjunctive or disjunctive connections [108], i.e. whether the nodes are connected by AND or OR logic [140]. An AND gate requires that all steps (leaf nodes) be complete before the attack is complete, and

an OR gate requires that at least one of the steps is complete before the parent node can be considered complete. The tree can be further refined by assigning more granular logic such as SEQUENTIAL AND or SAND [46]. Leaf nodes can be assigned Boolean (such as possible or impossible) or continuous (such as cost) values.

The structure is acyclic, requiring a root (attack goal), and is directional, which is significant. From a design perspective, a top-down approach, where an attack goal is first identified followed by all subsequent methods of achieving the goal, early in the development lifecycle, is recommended [148]. From a testing perspective, however, this is challenging because of the black box nature of security testing. Since, in this case, the system already exists, the tree here is built bottom-up tracing from leaf to root, based on observable entry points and subsequent behaviours when probed, leading to potential attack goals: the very process that penetration testing is based on. Note that although the *structure* is acyclic, the *process* of security testing as presented here (requiring multiple iterative test runs) can be considered cyclic.

As there are no real-world measures for detection of security incidents on a vehicle, the primary method of validation remains domain expert input (such as is the case for building the tree from bottom up [158]) and data from practical applications; this best practice has been used by others [30].

There are several assumptions usually inherent in the formation of these trees [5]:

- A defender must have vulnerabilities
- An attacker must have enough resource to exploit these vulnerabilities
- An attacker must gain some benefit from the attack

These assumptions, however, only cover what Wolf, Weimerskirch, and Wollinger [160] describes as a *rational attack*; where the cost of the attack does not exceed potential gains. In order to take this into account, a number of studies support the use of "holistic" methods, in that the results of these methods have more value when put in context of the whole [12], [48], [73], [128]. This perspective is by no means straightforward due to the complexity of automotive systems with increasingly blurred system boundaries.

Although numerous examples of Bluetooth weaknesses have been expounded upon (see Chapter 3.2), they have not generally been set in the context of the automotive system, where the system is opaque, the specifications absent or loose and information regarding implementation is scarce.

In conclusion, the priority of such a method would be a methodical approach laying the groundwork for what Nilsson and Larson [116] describes as *defence in depth*: a multi-layered approach that covers as many attack surfaces as possible and addresses multiple, potential attack scenarios.

4.2 PENETRATION TESTING

One of the foundations of penetration testing is the flaw hypothesis approach [109] which encompasses activities around generating and confirming potential flaws after a study of the system under test [109]. The results of the actions taken to carry out these activities can be considered deterministic based on system implementation, configuration and state [70].

This makes coming up with an accurate figure for test coverage challenging, as the above implies a very large (possibly infinite) number of results based on any number of implementation-configuration-state combinations (Figure 5) which are ever expanding in nature and scope.

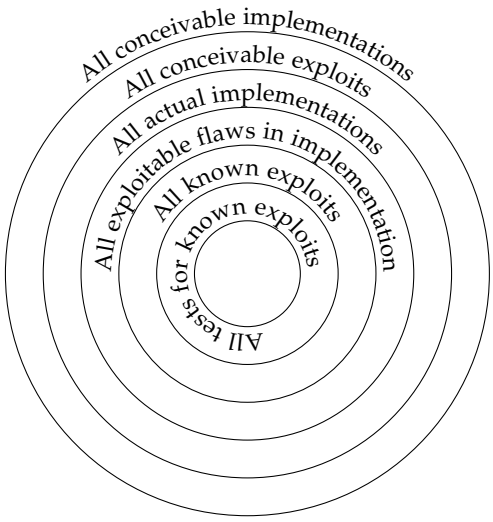

Figure 5: The security testing problem space

Although enumeration of weaknesses and vulnerabilities are possible through this method, there are limitations:

- Firstly, proving the complete absence of insecurities in an implementation is not possible, as tests only ever expose a limited subset of vulnerabilities [59]. It can only be stated that under certain abuse cases, these flaws were not present and that this is acceptable security;
- Secondly, a method that does manage to end in exploitation may not be the only method that does, however, the underlying flaw is exposed and can be addressed for that particular method. Abstracted patterns of this method can also be extracted to test for similar weaknesses through other vectors. For example, a buffer overflow exploit is mechanically the same whatever the system.
- Finally, there are concerns with generalisation of a system since such testing tends to rely on implementation. As automotive production lines are standardised, we reasonably assume that if the Bluetooth stack in a vehicle is flawed in some way, this same vulnerability may appear in some other vehicle of same age, make, model and software version (of which there may be millions). Furthermore, reuse is a common approach to reducing cost in the industry [125], and as such flaws could be replicated even in newer models. Even vulnerabilities that have been patched in more modern embedded systems may be present in newer vehicles, as software in an automotive system is updated less frequently [115].

In addition to these drawbacks, penetration testing is expensive. This is because there is one irreplaceable factor that cannot be automated: the human element.

Although the penetration testing process might involve the use of automated scripts, software or hardware tools and frameworks, the test generally hinges on the expertise and experience of the testers themselves. Methods used are usually not prescribed, although some, such as identifying machine addresses, are more common than others. As evidenced by the reports of vehicle hacking, the reason for the lack of automation lies in the unique nature of the human mind, which is able to think laterally to bypass otherwise sophisticated countermea-

sures, and work creatively to think within constrained environments (such as is the case with embedded systems).

Despite the lack of a coverage metric and the expense, penetration testing is still valuable. Firstly, considering the need to adopt the mentality of an adversary when using this methodology, system designers may not be the best persons for such an effort, as cataloguing all implicit assumptions (especially from a malicious viewpoint) made during system development is extremely difficult. Hence, an external security testing process [124] based around an adversary's perspective could provide insight into potential weaknesses that may not have been considered by system designers. Secondly, evidence compiled through the testing process can be used to provide a security assurance case, subject to the limitations described above.

4.2.1 *The Black Box Approach*

Ultimately, in practical terms, the aim of any penetration testing exercise is to assess the real-world security of a system from the viewpoint of an attacker, by not just discovering vulnerabilities, but sometimes actively trying to exploit them. Testing could take the form of white box testing (where all necessary information including documentation, designs, diagrams and source code is made available) or black box testing (where only investigation of the system is authorised, but no further information is given). There are several aspects to consider when using either approach. Having all the needed information and data saves time and allows for scrutiny of code or documentation that might otherwise be left unassessed or unconsidered. Conversely, black box penetration testing may allow for a more realistic assessment of the system, as testers would have only the same access as a potential attacker. Finding the same flaws as those that would be found through white-box testing, however, would likely take more time. Nevertheless, it is important to note that either approach will only ever provide points for improvement at that point in time. Neither approach is in itself a guarantee of future security.

In this thesis, the black box approach was taken with regards to security evaluation, both because information on the system is restricted due to trade confidentiality, and because it reflects reality on three fronts:

- Firstly, original equipment manufacturers (OEMs) traditionally buy in components or subsystems from Tier 1 suppliers. Although OEMs have access to the original specifications (which may or may not have included security considerations), they will have at best only partial sight of what has been implemented due to commercial sensitivities.
- Secondly, criminals who wish to investigate the system would also face similar constraints (although possibly less stringent owing to the assumption that they do not operate within legal or ethical bounds).
- Finally, the attacker experiences what [70] describes as "uncertainty of state", in that knowledge of the system to be tested is usually incomplete. This accurately reflects our own experience.

In response to market demands, there has been a move towards more complex processes within development of embedded systems software, based on the added functionality surrounding comfort and convenience. This is most apparent in vehicular infotainment systems. Many of these added features are based around existing IT technologies such as Bluetooth. Because of this, certain principles could also be adapted from existing security testing methods [147], such as the use of "librarian" methods (executing a set of scripts that represent known exploits) to find common errors.

4.2.2 *Penetration Testing Execution Standard*

The penetration testing process requires a degree of flexibility and adaptation depending on scope, objectives, resources available, industry practices and the nature of potential flaws. Technical standards already exist, but come with various barriers to widespread adoption, including the high information needs about testing environment (ISO/IEC15408), or only addressing part of the whole challenge such as is the case with the ISO/IEC 27001 standard on information security management [90]. Nevertheless, recognition of the fact that there had to be a broad common approach resulted in the proposal of the Penetration Testing Execution Standard (PTES) [122]. Although named as such, PTES is not a formal standard, but rather a technical methodological guideline to provide optimum test coverage; this,

however, means that there is still a lack of a globally accepted methodology for penetration testing.

The advantage to these guidelines is that, although there is lack of a concrete consensus on what constitutes a flaw, weakness or vulnerability, the process can, at least, be loosely categorised and ordered around:

1. **Scoping**, whereby scope, objectives, aims and ground rules are established and agreed to between all stakeholder parties,

2. **Information gathering or reconnaissance**, which involves a background study on the test subject looking for all possible weaknesses, whether that be through user, developer or system documentation, publicly available manuals, source code or results of previous testing. This process can be passive (listening for information) or active (probing in order to acquire a response) depending on the aim of the test. An example of this could be profiling the Bluetooth module or chip, where the NAP and UAP information could be used to identify the manufacturer (a list is publicly available in IEEE's Standards Register [77]).

3. **Formation of vulnerability hypotheses**, which involves constructing hypotheses of possible vulnerabilities in the system, determined via study of background information gathered as well as by looking at abuse cases,

4. **Threat modelling** which involves the generation and modelling of possible threats based on potential vulnerabilities identified,

5. **Testing and exploitation**, which is used to establish the presence of the vulnerability, determining the nature of the flaw, whether it is repeated through the system and its security impact,

6. **Clean up and report**, which involves creating or collating recommendations for discovered flaws. These are then presented, along with a cleanup of the system to ensure that no inadvertent flaws from penetration testing (such as malware or backdoors) are left behind.

The categorisation above is not necessarily a step-by-step process. Each stage (or series of stages) can be re-iterated as needed for the

system or component being tested; threat modelling for example may uncover a breadth of testing that might not necessarily be in scope, which would mean re-visiting or re-writing the scope in order to match any constraints more accurately.

Although a mass of information has been acquired regarding tools, techniques and to some extent, motivation (in terms of attacker goals and what they hope to gain), the issue of prioritisation and the combination of circumstance that would result in the use of this tool or that attack method can be covered by scenario building. In this research, the latter is covered through the building of attack trees (see Section 4.1).

4.2.3 *Conclusion*

Recall that penetration testing results can be considered deterministic based on implementation, configuration and state. Despite this, results can neither be predicted nor calculated when a black box perspective is employed, which made vehicle-based experimentation empirical. Nonetheless, there is value in carrying out such research. The complete list of vulnerabilities may be unenumerable, but corrective action to address an observed vulnerability would reduce that list by at least one, and the information from the testing process contributes to a security assurance case (explored further in Chapters 6 and 7). The results could also provide the owner of the system with information for improvements or verification of current configurations, either via a security assurance case or through further work involving formal methods (see Chapter 8).

5

IMPLEMENTATION

A software tool was designed and created to aid in the semi-automation of the systematic evaluation process (see Section 5.1).

This tool is the implementation and embodiment of the concepts and methods described in Chapter 4. It is designed to be used in the implementation testing phase of any development cycle. This can be as early as prototype phases, right through to production line or assembled devices. At this stage of development, the developed tool helps the systematic search for weaknesses. A weakness can be defined as a state that increases the risk of an attacker accessing or exploiting a vulnerability [71]. Recall that a vulnerability can be defined as a specific instance of a weakness (such as an insufficiently protected buffer that allows for a buffer overflow). Tool development, including workflow, the general algorithm and key features are discussed in Section 5.1.

As the systems under test are black boxes, any measure of code coverage with regards to the system is not possible (see Section 5.1.5). However, depending on the results, countermeasures developed to address these weaknesses can be construed as (partial) assurance (see Section 5.1.4), which can be described as a measure of security that allows for reliable operations [61]. There is also no abstraction within the tool when testing the target system as we are testing the implementation of Bluetooth technology deployed in a vehicle. The tool was then benchmarked against systems with known behaviour (Section 5.2).

5.1 TOOL DEVELOPMENT

The proof-of-concept tool created is an extension of the concepts embodied by various other proof-of-concept, pre-alpha and beta Bluetooth security testing tools created since 2003 (Table 10).

Some examples of these early tools include *redfang* [156] (a proof-of-concept tool created in 2003 for brute-scanning), *CarWhisperer* [167] (a small tool created in 2005 to scan for manufacturers that implement

hardcoded PINs and use that to connect to and inject audio into or record audio from a vehicle), or *Bluesnarfer* [106] (a tool created to exploit a vulnerability discovered in 2003, using AT commands in order to extract phonebooks from susceptible mobile phones). The most recent release of *nOBEX* [113] can be construed as the most relevant as it is directed at automotive headunits. However, functionality is currently limited to fuzzing and they make no claims as to automation.

Additionally, many of the features built by the author into the developed tool (see Section 5.1.4) were only publicly available as conceptual techniques in literature (such as the BlueSnarf++ [111] attack). Still others had limited functionality due to use of old libraries, or were built for Bluetooth implementations that are no longer prevalent (such as BTBrowser [11]).

The methods used to extract information are part of known literature. Use of `AT` commands, for example, forms part of the *Bluebug* attack [4], directory transversal using OBEX FTP is also known as *BlueSnarf++* and is described in [111], whilst flooding receptive L2CAP ports (also known as *Bluesmack*) was an attack identified by the Trifinite group [149]. Although the methods of attack in themselves are not novel, there are several aspects of our tool that contribute to the research as presented in this thesis:

- The test suite itself is more comprehensive than any of the tools listed in Table 10. It covers the possibility of using it for surveillance, fuzzing, sniffing, DoS and direct data access in one tool. Furthermore, the attacks that the tool performs have been systematised using attack trees (see Section 5.1.3), with the attendant benefits (see Chapters 2 and 4.1).

- There are parts of the created tool that specifically deal with the automotive diagnostics interface and the connection to a Bluetooth-enabled vehicular aftermarket device (see Chapter 7). These features were not found in any of the tools in our survey above.

- Finally, the tool was also created such that semi-automatic severity classifications based on the SAE J3061 standard (see Table 1 in Chapter 2.2) could take place (see Chapters 6.6 and 7.5).

Note that the application to a vehicle (with black box systems) and the systematic evaluation to establish a baseline security state, was the

Table 10: Tool and technique survey adapted from [17], [44], [66]

Attack type / *Tools*	Surveillance	Range extension	Obfuscation	Fuzzing	Sniffing	DoS	Malware	Direct data access	MITM	*Description*
bluez suite	x		x					x		Tools from the official Linux Bluetooth protocol stack
bluecasing	x									Act of finding devices to intrude
redfang	x									Tool to find non-discoverable devices
bluesniff	x				x					Bluetooth war-driving utility
BlueSniping		x								Long range version of Bluesnarf
Vera-NG		x								Range extension tool
bluetooone		x								External directional antenna attachment for range extension
BluePass				x						Bluetooth fuzzing tool
BlueSmack				x		x				The act of flooding open ports using Bluetooth
BlueStab				x						Act of fuzzing by using malformed Bluetooth device name
Tanya				x						Tool for flooding L2CAP ports with large packets
CarWhisperer				x				x		Tool to find and connect to devices with hardcoded PINs
BTStackSmasher				x						Fuzzing tool
Ubertooth					x					Bluetooth sniffer
Merlin					x					Bluetooth sniffer
CommWarrior							x			Symbian Bluetooth worm
Cabir worm							x			Symbian Bluetooth worm
BlueBag							x			Covert infection of device
BlueSYN						x				Act of using and sending crafted packets to cause DoS
BlueJacking				x		x				Sending of unsolicited messages to Bluetooth-enabled device
vCardBlaster				x		x				Act of flooding remote device with either valid or malformed contact information
BTcrack								x		Bluetooth PIN cracker
Bloover								x		Tool that implements Bluesnarf
Bluesnarf								x		Theft of information from a device through Bluetooth
BlueSnarf(++)								x		Directory traversal through use of FTP
BlueBug								x		Act of pulling information from a phone using AT commands
BlueSpooof			x						x	Bluetooth address spoofer
spooftooph			x							Bluetooth address, name and class spoofer
Printer-MITM									x	Technique used for MITM against Bluetooth-enabled printers
obexstress						x		x		Fuzzing tool used to send malformed packets through OBEX channels
l2ping						x				Tool for flooding L2CAP ports
nOBEX				x						Tool used to fuzz using phonebook synchronisation profiles

focus of the tool, rather than the identification of novel attack methods. This was due to time constraints. However, the tool is modular and extensible. As the attack tree grows, the number of techniques encoded in the tool might result in the elicitation of a novel attack method based on any combination of increasing attack tree nodes.

The proof-of-concept tool was developed using Python 2.7 on a Kali Linux system. It requires the *BlueZ* Bluetooth stack (this tool was developed and tested using *BlueZ* 5.x [15]) and the Bluetooth Python extension module *Pybluez* [166]. The tree structure and the tree search facility is enabled by the *treelib* library [37]. A full list of required Python modules and libraries can be found in Appendix A.1.

The chip in use at the local interface is a Cambridge Silicon Radio (CSR) Bluetooth 4.0 chip. Other Bluetooth adaptors were tested, but other than the chips manufactured by CSR and Broadcom, most test runs exhibited anomalous behaviour (such as being able to discover devices, but not able to query device information). The tool was also tested on a virtual machine. However, the virtualisation of the Bluetooth adaptor was a source of many problems (such as not being able to discover device addresses), and so development was restricted to a physical machine.

The tool does not require any extraneous or special processing power or memory beyond that found in a standard Linux machine, although a more powerful machine running the tool is advantageous. This is especially relevant when testing a vehicle using aftermarket devices that are attached to the OBD-II port, as vehicles generate a lot of data very quickly. Root access on the machine that runs the tool is necessary, as that is required by some of the local libraries the tool makes use of.

5.1.1 *Workflow*

The workflow of the tool follows pre-determined attack trees (see Chapters 6 and 7). The content of the attack tree depends on the attack goal (synonymous with the ultimate aim of an attacker).

The initial construction of the tree was manual, with test cases drawn from Bluetooth exploits in other areas (Table 10) as well as descriptions of known vulnerabilities in Bluetooth in other domains (Table 4). There is a one-off time and effort cost in building these trees,

however, once constructed, the trees (or appropriate subtrees) can be reused or expanded upon in future test runs.

As the tool runs through the test suite, there are points where the user has the option to skip or redo a particular test with different parameters. Also for the purpose of test case efficiency, the tool runs through checks (for example, for open ports and certain profiles) before continuing. This is so that time is not wasted running an extensive test - such as the port scan - if the device Bluetooth address is not found or is invalid.

Logs are generated as each of the attack steps are run. The logs saved are in either text or comma-separated values (`.csv`) format. The text file format is used to simply log output as is displayed on the terminal. The latter case is usually when there is large textual output that may be of value to a tester, but has no specific set or list of values to be mapped into a `csv` file. At the end of the test, the logs (including historical logs) are collated locally in an appropriately labelled and time-stamped folder.

An important part of the output is the populated tree structure. This contains information about the tests run, with information on which logs to view should that particular leaf node be of interest, as well as suggestions (in the form of subtrees) for tests that have failed, or returned no results. This is performed by referencing a pre-built master tree which contains a master set of known possible actions for each of the attack steps. Where the populated tree contains a `NULL` indicator, the tool searches the master tree using width-first search and displays the appropriate branch.

5.1.2 *Conceptual Architecture*

The general architecture of the tool is shown in Figure 6. At the top level, the only input that is required is the Bluetooth address of the device in question. This address can be acquired via scanning (also incorporated into the tool) or from manual methods such as reading a label. Further generation of input values is dependent on the results of the tests preceding it, hence the sequential nature of the tree. Examples of this are the tests that require a certain service profile (such as the OBEX FTP) to be available. The tool checks for the presence of such profiles, before running subsequent tests.

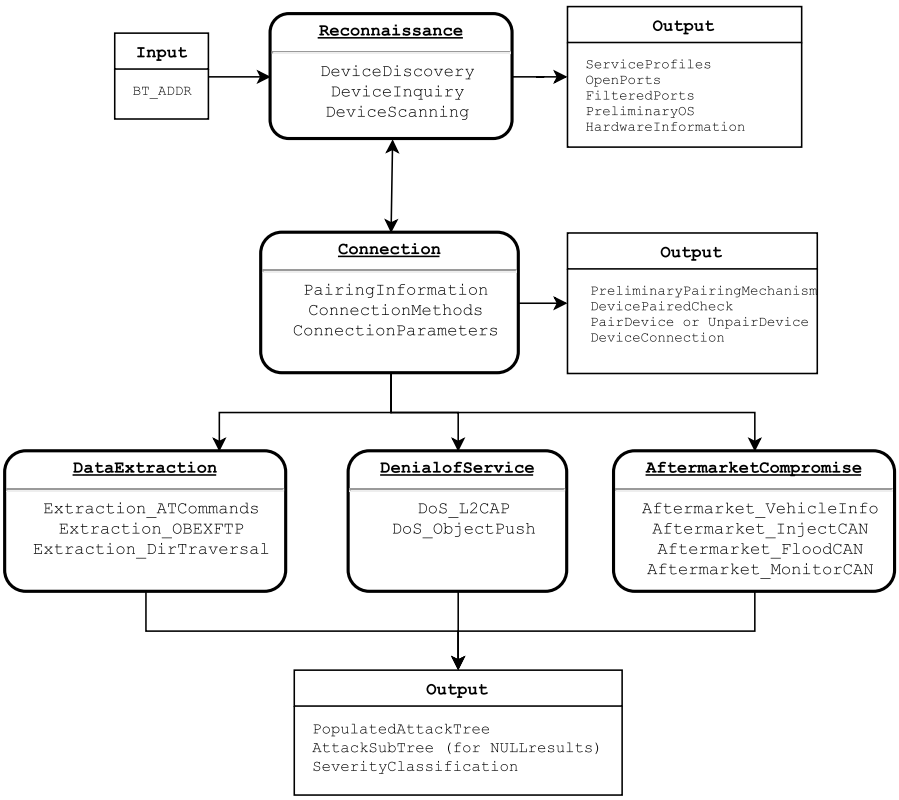

Figure 6: General architecture of the Bluetooth enumeration tool

There is also an element of fuzzing, especially with regards to the data exfiltration tests (for example, by entering AT commands to see what output can be acquired). However, these tests are meant to be a qualitative indicator that the system is susceptible to such attacks; further work would be required to ascertain the exact set of commands that would elicit a desired response.

There are several points where manual intervention is required, for example, to enumerate the type of SSP association model that might be in place (this is discussed further in Section 5.1.4.2). Where information about a system cannot be found, or manual intervention did not produce anything, suggestions from the appropriate sub-tree (taken from a master attack tree) would be amongst the outputs.

Other outputs include logs of the service profiles, the port scans, results of AT commands, results of denial of service floods, or output when attacking a vehicle through an attached aftermarket devices. The attack tree that the tool follows, populated by results of the test (or where appropriate, the name of the log file) are also amongst the logged outputs.

The tool currently performs predefined actions as described by the attack tree. The modular design of the tool means that as more information or data becomes available and is added to the tree, more actions can be added to the tool, so that it runs in tandem with the attack tree. This applies only to where the action required is software-based rather than physical (such as stealing a device).

5.1.3 *Algorithm*

The process of investigating the headunit (Figure 7) begins at the highest `SAND` gate and travels down each node of the tree, using the appropriate logic gates, until it reaches the leaf nodes.

Following this, (again dependent on the logic gate), the tool carries out the necessary testing steps, recording and outputting data to the appropriate test run. Vehicle data here refers to any data that is available from the vehicle, including personal data, vehicle-generated data, or data about the vehicle itself.

Investigation of the vehicle through attached aftermarket devices follows largely the same process, with reconnaissance performed in the same fashion (although in this case, the reconnaissance involves learning about both the aftermarket device and the vehicle to which it is attached). Attacks are then performed on the car (or its components) depending on the logic gates of the attack tree, with vehicle data output to logs.

5.1.4 *Key Features*

The features of the tool can be categorised broadly into:

- **Reconnaissance**, which can be defined as a survey of the system's existence, configuration and capabilities (Table 11),
- **Connection attributes**, which includes information on pairing mechanisms, transmission sizes and connection state (Table 11), and
- **Attack goal**, which encompasses methods that would allow the realisation of the attack goal (Table 12);

```
input  : Predefined attack tree
output : Vehicle data
initialization;
for AttackGoal do
    foreach AttackTreeBranch in order do
        foreach AttackTreeLeaf do
            if AttackTreeLeaf is OR then
                while attack fails do
                    AttackSteps on vehicle;
                end
                Record vehicle data;
                if no vehicle data then
                    Display AttackSteps and AttackTreeLeaf;
                else
                    Populate AttackTreeLeaf with vehicle data;
                end
            else
                perform all AttackSteps;
                Record vehicle data;
                if no vehicle data then
                    Display parent nodes with children for AttackSteps;
                else
                    Populate AttackTreeLeaf with vehicle data;
                end
            end
        end
    end
    for AttackTreeLeaf do
        if has vehicle data then
            for AttackTreeLeaf do
                Display AttackTreeBranch;
            end
        else
            for Empty AttackTreeLeaf do
                Display AttackSteps;
            end
        end
    end
end
```

Figure 7: Algorithm used for testing the Bluetooth interface [33]

Table 11: Proof-of-concept tool features (expanded version from [33])

	Feature	*Method*
Reconnaissance	(A) Discovery of 'discoverable' device addresses	Inquiry scans for available Bluetooth addresses
	(A) Discovery of 'hidden' devices	Brute-force scanning (incrementing the address bits by one before sending an inquiry). Requires preknowledge of the first three bytes of the Bluetooth address (OUI) - or other address bytes to be feasible.
	(A) Determination of device manufacturer	Using the OUI to scan through a database of stored OUIs (from IEEE's standard register [77])
	(A) Determination of Bluetooth chip manufacturer	Using device information supplied by *bluez*
	(S) Retrieves FCC ID information	Retrieves relevant Federal Communications Commission (FCC) web link if ID is known by user
	(A) Determination of service profiles offered by device	Using the Service Discovery Protocol (SDP)
	(A) Preliminary indication of device operating system (OS)	Using indicators in discovered service profiles
	(S) Determination of whether device uses legacy pairing	Checks Bluetooth version (version 2.0 or before means that legacy pairing is in use, whilst version 2.1 or above means that use of either legacy pairing or Secure Simple Pairing (SSP) is possible)
	(A) Determination of open ports	Sending information to all possible RFCOMM and L2CAP ports and awaiting the appropriate responses
	(A) Determination of filtered ports	Sending information to all RFCOMM and L2CAP ports and filtering for specific error messages
Connection Attributes	(A) Determination of pairing status	With reference to local paired devices list
	(S) Pair or unpair the device as appropriate	With reference to local paired devices list, subject to appropriate authentication
	(S) Spoof a device	Calls to installed *spooftooph*[43] package
	(A) Checks for presence of OBEX File Transfer Profile (FTP) service	With reference to discovered service profiles
	(A) Checks for presence of OBEX Push Profile (OPP) service	With reference to discovered service profiles
	(A) Determines maximum transmission unit (MTU) for open L2CAP ports	Sending increasing size of packets until Bluetooth error `90, message too long` appears

(A) = fully automated, (S) = semi automated, requires manual intervention

Table 12: Proof-of-concept tool features (continued) (expanded version from [33])

		Feature	*Method*
Attack Goal	*Data Extraction*	(S) Attempted extraction of information from headunit	Using "attention modem" (`AT`) commands through open RFCOMM ports
		(S) Attempted extraction of information by browsing headunit filesystem	Mounting the filesystem on a "Filesystem in Userspace" (FUSE) based filesystem type (if OBEX FTP exists)
		(S) Attempted extraction of information	Using the *dot-dot-slash* (`../`) attack to attempt directory traversal beyond the given restricted directory (if OBEX FTP exists)
	DoS	(A) Attempted denial of service	Flooding open L2CAP ports with L2CAP echo requests
		(S) Attempted denial of service	Repeated data push through OBEX channels
		(S) Attempted denial of service	Pushing of malformed data through any RFCOMM channels
	Vehicle Compromise	(A) Extract vehicle specific information	Vehicle information based on `AT` commands sent to an attached wireless Bluetooth-enabled OBD-II device
		(S) Attempted extraction or denial of service	Through injection or flooding using OBD-II protocol messages (see Chapter 7). User specifies parameters such as type and number of messages and time intervals
		(S) Attempted vehicle compromise	Through injection or flooding using raw pre-determined CAN messages (see Chapter 7). User specifies CAN header ID, CAN data payload, number of messages to be sent and time intervals
		(A) Vehicle data extraction	Passive monitoring of all exposed CAN buses on the OBD-II port
Outputs		(A) Scan logs	Written to CSV or TXT files and collated at the end of the test run
		(A) Populated attack tree	Displayed and logged with results of the test run
		(A) Subtrees	Where test results have not been found or entered, denoted by `NULL`, appropriate subtrees (found using width-first search) will be displayed and logged.
		(S) EVITA Severity classifications	For each result in combination with answers to tester queries, a severity classification based on the "privacy" and "operational" categories (see Chapter 2) is given.

(A) = fully automated, (S) = semi automated, requires manual intervention

- **Reporting**, including logs of all data gathering and tests run (Table 12), parameters chosen by the tester (where appropriate) and the severity classification (see Chapter 5.1.4.1)

Each of these categories is an individual component (with the individual attack steps forming sub-components). The attack goals in this case were two examples of the attack classification as described by [44] and are in line with the general accepted security testing goals of violating confidentiality, integrity or availability (CIA).

This structure allows for different permutations (depending on the attack tree desired), and for extensibility; new attack methods can be added as steps (as sub-components) within each module. New attack goals that come within test scope can also be added to the tool as a different module. Since the beginning of every security test begins with an inspection of the system, the reconnaissance module can likewise be re-used at the beginning of every test run.

5.1.4.1 *Severity Classification*

Based on the EVITA severity classes as described in Chapter 2.2, the diagnosis element of the tool assists in classifying severities of the various data points discovered.

Although there are four classes, only the privacy and operational classes were considered. This is because the safety classification would need safety analyses (and the attendant processes by which to judge its severity is outside the scope of this thesis). The financial category pertains to two aspects of financial loss. Firstly, loss through financial transactions within the vehicles, a feature that is not yet widely deployed and not present on any of the vehicles tested. Secondly, the financial rating could also be due to severity of loss through theft of the vehicle. However, there is no definitive real world detection measure or tool to determine whether a (Bluetooth) vulnerability led to this theft, even should the vehicle be recovered.

This part of the tool is semi-automated, in that there are several questions for the user to answer (based on manual observations) in conjunction with the results of the appropriate testing process. A severity level (S0-S4) is then given to each of the findings (labelled 'ACT' for actual). Furthermore, a potential severity level (based on worst-case scenarios should an attacker be able to carry through to exploitation) is also given (denoted by 'POT'). All of the above is tab-

ulated with a rationale for each of the ratings (Figure 8). For specific examples of questions asked and the results, please refer to Chapters 6 and 7.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	Address XX:XX:XX:00:21:CC	Sp2,So0	Address unique to each device
POT		Sp3,So0	Vehicle Tracking Possible
ACT	OUI: Example	Sp1,So0	Anonymous data acquired
POT		Sp1,So3	Further recon may lead to operational S3
ACT	OS: NullOS	Sp0,So0	No data acquired
POT		Sp0,So0	
ACT	Service port no:1	Sp1,So1	Synchronisation services found
POT		Sp3,So1	If unauth. sync successful --> privacy Sp3

Figure 8: Example EVITA severity classification table

This process is incorporated into the tool to allow for some guidance as to the security assurance required in the context of EVITA and J3061 (see Chapter 2.2, Table 1), which, at this time, is the only automotive specific cybersecurity standard. Improvements to the system could be prioritised based on how high the actual or potential severity levels are. Because these trace back to the attack tree, the hierarchy of the attack tree can then be used to locate the root problem to be addressed. An example of this is given in Chapter 8.3.

5.1.4.2 *Automation*

The tool is semi-automated, in that many aspects of the test suite do not require manual intervention. This is true of the majority of the reconnaissance and connection attribute determination features. However, some manual decision making is required when performing any of the attack goal tests. Automation is difficult in these cases as the target system is a black box, and the search for weaknesses in such an environment comes with a large number of sequential decision making issues [70]. Indicators regarding level of automation for each of the features are given in Table 11 and 12.

Subsequently, a key consideration during development of the tool was the question of which aspect of the test suite could be automated. There were some aspects where non-trivial development was required, when it would have been simpler to observe the target device (such as which of four SSP mechanisms were in use during pairing), and others where manual intervention was required as the information is generally held physically. An example of the latter is the device's Federal Communications Commission (FCC) ID. This ID is

granted when a device requiring any kind of radio frequency communications is to be sold or imported into the United States. Manufacturers are obliged to supply electromagnetic capability information, block diagrams, functional tests and safety analyses (amongst others). Not all of the information is publicly available. However, it provides valuable information on the attributes of wireless communications, and is usually printed on a physical label.

Some manual observation also plays a large part when running through the test suite. The denial of service test for example was tested against a phone call made, to see if there was any discernible disruption in service or connection. Whilst the tool records the number of packets sent and transmission sizes, there was no non-trivial way to quantitatively measure connection quality at the same time.

In summary, the tool, whilst being only semi-automated, provides a significant head start with regards to establishing the security state baseline for the target system. Furthermore, the attack tree methodology underlying the tool also provides for a traceable and systematic set of results.

5.1.5 *Input Domain Coverage*

The Bluetooth and OBD-II interface as explored in this thesis (see Chapters 6 and 7) is of indefinite size and complexity, since no information is available regarding specifications or execution paths. This presents problems when looking at systems testing, and necessitates careful consideration of input values and the sequence thereof.

This is similar to the problem of automation as discussed above, and in a complete black box, testing for non-functional properties such as security requires manual intervention and observation in order to complete the test suites since execution paths are unknown.

The second problem within the testing space is that results of the same test do not always present themselves, since the front end (with the user interface) could be a separate on-board unit to that of the back-end providing the software. Thus the front-end software could be programmed to ignore exceptions, may not recognise exception handling alerts or may call other (unknown) functions in the event of invalid input. The back-end unit may also respond to errors in a similar fashion, and of course, implementation differs from model to model of vehicle.

Therefore, although boundaries are commonly used in testing [157], they are incredibly difficult to establish in this case. There is no knowledge of minimum or maximum values (and therefore no knowledge of what the extreme inputs might be), and tests to gather data on such may only result in notices based on the Bluetooth standard rather than from the underlying operating system. An example of this is sending malformed characters through open RFCOMM ports. Sending a value with a length of one byte or 10 bytes resulted in the message "forbidden" in one vehicle, but in other vehicles there was no response whatsoever.

In the case of the thesis, the attack trees are used as the test set by which to evaluate Bluetooth interfaces in vehicles. The aim of this set is to reveal undesirable behaviour (i.e. the attack goal). Since there is no knowledge with regards to the controls, outputs or code interfaces to be stimulated, we can only infer based on known vulnerabilities in Bluetooth in other domains. Within this set, the subsets are each of the main branches of the tree (reconnaissance, or connecting to the target system). As long as each individual attack test is performed in accordance with the appropriate logic and is successful, we can assume that the undesirable behaviour exists. Note that the converse does not hold true (in that if no undesirable behaviour is found, there cannot be any guarantees or assumptions as to its existence).

Code coverage is not feasible in this study because of the black-box nature of testing and complete coverage is typically impossible. We make no claims that the attack trees used here are complete, but is a representative sample that would allow us to achieve knowledge regarding insecure, and therefore undesirable, behaviour.

5.2 BENCHMARKING

The tool was tested against devices with known behaviours. An example of this is the Nokia 6310i, known to be susceptible to *Bluesnarfing* (the theft of information from a mobile device [106]). Testing was also performed on other, more modern, mobile phones since these devices have known responses to probing. This was performed both to verify functionality of the key features, as well as to gather more information on what could be included in the test suite.

The second set of tests were performed against a miscellany of small embedded devices (including keyboards and aftermarket OBD-

II dongles) to provide another benchmark (this time from less sophisticated devices, but with less transparency as to what might be contained therein).

Although the functionality of the proof-of-concept tool was verified, direct correlation with insecure behaviours as described in early Bluetooth security literature was difficult as many of the benchmark Android phones used newer forms of Bluetooth. These phones all issued notifications and asked for user confirmation, even with the phone already paired to the test laptop. Furthermore, functionality found in smartphones (and the other embedded devices) could differ from what is found in vehicles (as versions of Bluetooth profiles may differ). To truly test whether the presence of at least one vulnerability was adequately addressed, a Nokia 6310i (introduced in 2002) was also included in the test set. Further benchmarking was resource constrained. As many of the reported vulnerabilities are old (see Table 4), not every device with confirmed vulnerabilities (some of which were also old) could be acquired in working order.

The summary of all tests performed on benchmark devices are given in Table 13. Since the tool is still at a proof-of-concept stage, it is neither exhaustive nor complete. Its primary purpose was to demonstrate and automate the systematic security evaluation of an automotive Bluetooth interface.

5.3 SUMMARY

The tool created for this thesis is intended to aid in a systematic security evaluation of the automotive Bluetooth interface. The tool follows a pre-defined attack tree in a semi-automatic manner and logs the results for further analysis. The attack trees were built using the attack goals of data extraction, denial of service and vehicle compromise

The test suites available in the tool were then benchmarked against various devices, including mobile phones. This was to verify the functionality of the created tool's features as well as to gather more information on what could be included in the test suite. The result was a software tool suite that could be used for systematic testing as described in the following chapters.

Table 13: Test results from benchmark devices

	OS	*OUI*	*Chip*	*Class*	*Bluetooth version*	*Observations*
Mobile phones	Android 5.0	LG	Qualcomm	0x5a020c	4.0	All reconnaissance proceeded without problems (but if phone was not paired, pairing would be initiated during port scans). Phones issue alerts even if pairing is denied by tester. All AT commands worked, file system was browsable (both subject to user authorisation). No directory traversal. No observable effects from DoS attacks.
	Android 6.0	HTC	Qualcomm	0x5a020c	4.0	
	Android 5.0	HTC	Mediatek Inc.	0x5a020c	4.0	
	Android 4.2	Motorola	Qualcomm	0x5a020c	4.1	
	Windows Phone 8	Nokia	Qualcomm	0x78020c	4.0	Reconnaissance proceeded as above. All AT commands worked (with user authorisation), but filesystem was not browsable or traversable (no FTP service). No observable effects from DoS attacks.
	Blackberry OS 7.1	RIM	Texas Instruments	0x7a020c	2.1	
	Nokia 6310i	Nokia	Nokia	0x500204	1.1	This phone contained a proprietary operating system, and was one of the first phones (and the first from Nokia) to have integrated Bluetooth. It was also a phone that had a confirmed vulnerability to Bluesnarfing [65] (pulling data from the phone using AT commands), which was confirmed by use of the proof-of-concept tool.
Embedded devices	*Device type*	*OUI*	*Chip*	*Class*	*Bluetooth version*	*Observations*
	Keyboard	No info	Broadcom	0x002540	3.0	These devices were very simple. The keyboard required SSP Passkey Entry, the dongle required a PIN of 1234 to be entered, the remote control had no requirements. Services offered were extremely limited (the dongle and keyboard only had one). Therefore no active tests could be performed.
	OBD-II ELM327 dongle	No info	CSR	No class given	2.1	
	Stereo Remote Control	Primax	CSR	0x240404	2.0	

6

EXPERIMENTAL APPLICATION: BLUETOOTH IN VEHICLES

Presented in this chapter are the experiments conducted with the developed tool (Chapter 5) on the infotainment units of eight real world standard production line vehicles. Experimental objectives and setup are presented in Sections 6.1 and 6.2 respectively. Results from the tested vehicles (which were all registered in the last six years and spanned seven different manufacturers with six different Tier 1 suppliers) are given in Section 6.3. Analysis of results is presented in Section 6.4.

There was no additional source of information regarding the Bluetooth implementation on these systems, other than what was publicly available through the owner's manuals. Only one of the vehicles was a test vehicle (the others being either a shared resource or private), and so the invasiveness of the tests performed had to be managed. These limitations are further discussed in Section 6.5. Finally, how this framework and the results thereof contribute to a security assurance case is explored (Section 6.6).

6.1 OBJECTIVES

This study had three objectives:

- Firstly, to establish a baseline security state of the vehicle, determining both capabilities and implementation specific details;
- Secondly, to extract information from and about the vehicle (in line with the first attack goal specified); and
- Finally, to cause denial of service disruption to the vehicle or vehicle's infotainment unit, in line with the second attack goal specified.

6.2 EXPERIMENT SETUP

Two attack trees (Figure 9 and Figure 10) with the attack goals of data extraction and denial of service was predetermined using Bluetooth techniques and attacks described in literature (see Chapter 3).

The experiments were carried out using a laptop with the proof-of-concept tool developed on a standard Kali Linux 2.0 distribution and a Cambridge Silicon Radio (CSR) Bluetooth 4.0 adaptor. For full details of implementation of the proof-of-concept tool used, please refer back to Chapter 5.

Vehicles were stationary and ignition was switched on, ensuring that it was within the Class 2 (ten metre) range as no antennas were used to extend range. If Bluetooth had to be enabled (which was usually the case) then this was performed before the tool was run. Information regarding each vehicle (with identifying features redacted) is given in Table 14.

For this study, primarily "legitimate" connections were used. Legitimate is defined here as a straightforward connection to the vehicle from a mobile phone, without any attempt to compromise the Bluetooth communications protocol. The reason for this was that the aim of the study was to explore Bluetooth as implemented on the vehicular system, rather than the Bluetooth protocol itself, since the security (or lack thereof) of the Bluetooth pairing mechanisms and the underlying protocols are well studied [32], [64], [65], [97]. A connection compromise would thus exacerbate the risk of malicious actions on a vehicle rather than introduce a new risk. The latter is especially significant since there are no access control policies on the vehicle, i.e. access to Bluetooth services implemented was not differentiated by user, but rather by device. Every device had access to every service (assuming compatibility, a property we considered out of scope during the testing process). There was no need to compromise the protocol to elevate privilege or gain access to restricted areas as might be the case with a more traditional computing system. The single exception to this was the use of a spoofed device (at the local testing interface), to observe whether the vehicle accepted the connection.

Table 14: General information regarding test vehicles

Vehicle no.	*Reg. Year*	*Vehicle type*	*Tier 1**	*OEM**
1 (Table 15)	2013	Small hatchback	A	I
2 (Table 16)	2012	Small hatchback	B	II
3 (Table 17)	2015	Sport Utility Vehicle	C	III
4 (Table 18)	2011	Small hatchback	B	IV
5 (Table 19)	2016	Sport Utility Vehicle	D	V
6 (Table 20)	2016	Hatchback	E	VI
7 (Table 21)	2012	Estate	F	VI
8 (Table 22)	2015	Saloon	D	VII

*OEMs and Tier 1 suppliers have been given an alternate denomination in order to anonymise the data

6.2.1 *Attack goals*

Two attack trees (according to two attack goals) were created for testing on vehicles. They are discussed below.

6.2.1.1 *Data extraction*

Data extraction from a vehicle through Bluetooth is largely an exercise in gathering as much information as possible. Although the most overtly valuable is information regarding the user of the vehicle, data from the vehicle itself (such as cornering speed) can be used to fingerprint drivers [49].

Other information that may be of use include the age of the technology (which may indicate legacy flaws), the chip manufacturer (which may point to analogous vulnerabilities or bugs), pairing mechanisms (where aspects such as using a fixed PIN might make a system more insecure) or other reconnaissance data that could enable targeted attacks. The attack tree used for this attack goal is shown in Figure 9.

6.2.1.2 *Denial of Service*

The denial of service goal involves flooding and fuzzing, as jammers were not available at the time of testing. The aim was to cause disruption to the vehicle or any component therein, with the primary target being the headunit (where Bluetooth implementations are usually deployed). Because of the nature of denial of service, and because the system under test is a black box, most of the results are based on

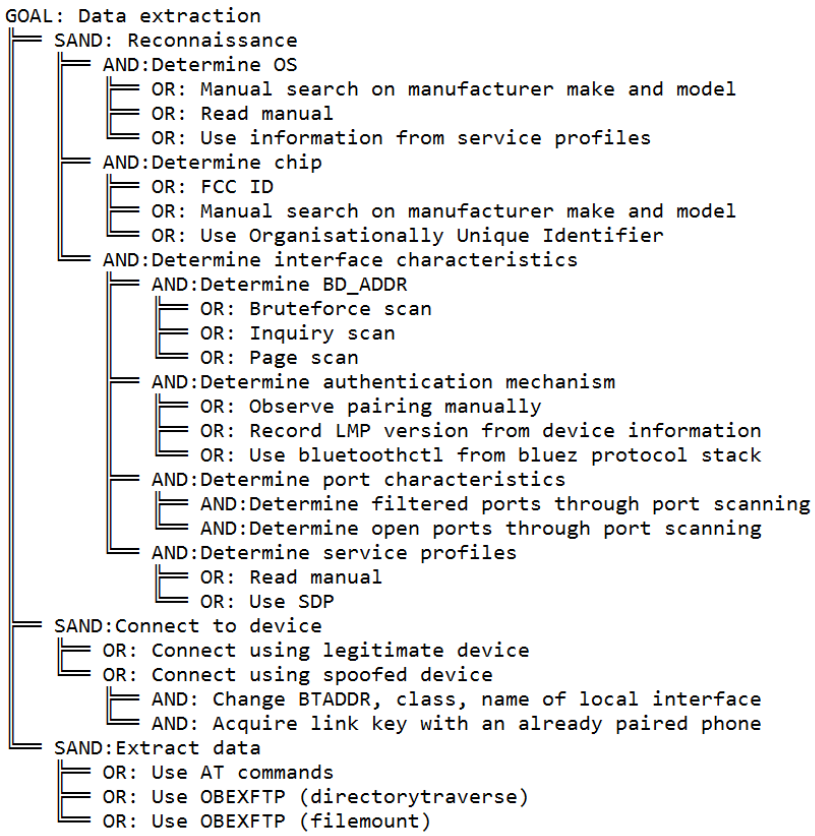

Figure 9: Attack tree with data extraction as an attack goal (adapted from [33])

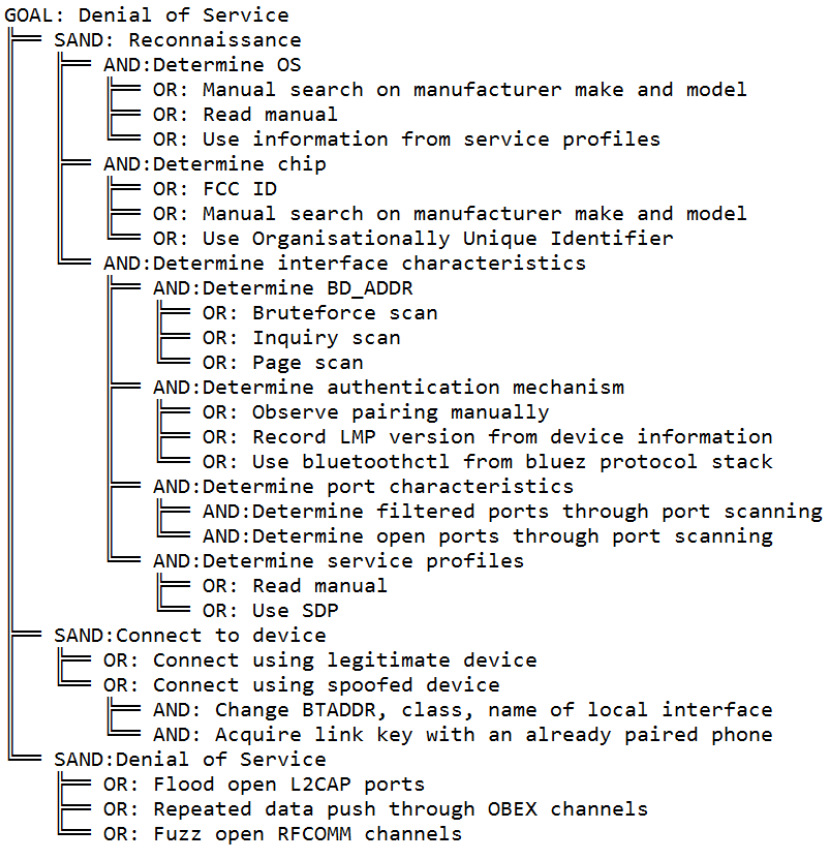

Figure 10: Attack tree with denial of service as an attack goal

observation whilst trying to perform normal actions (such as making a phone call), or on what might happen on the graphical front end, rather than anything being returned by the vehicle. The attack tree used for denial of service is shown in Figure 10.

6.3 EXPERIMENTAL RESULTS

A summary of results and observations is given for each vehicle tested in Tables 15 to 22. Furthermore, an overall summary of test effects of each of the attack goals is given in Table 23. Due to commercial sensitivity, identifying information for each vehicle has been redacted.

Table 15: Experimental Results: Vehicle 1 [33]

	Interface characteristics	*Observation*
	Address: xx:xx:xx:34:8A:2D **Version:** 2.0 **Class:** 0x340408 **Services:** HFP, SyncML Server, A2DP, AVRCP, PBAP (Client), OBEX OPP, MAP MNS **Open ports:** RFCOMM 1,4 and L2CAP 1,3,23,25	Bluetooth version 2.0 means that vehicle is using legacy pairing exclusively. Vehicle produces dynamic 6 digit PIN. Device class interprets to an audio/video hands-free device Services include a Synchronization Markup Language (SyncML) Server (for phonebook, message and calendar information synchronisation). A SyncML client could be used to extract personal information that is stored on the vehicle (although connections through this were unstable - no information was found).
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	Responds to all `AT` commands sent on channel 4 with `AT+BRSF=39`. Commands on other channels end with errors such as `103, Software caused connection abort`. No OBEXFTP, so filesystem cannot be accessed through this vector.
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Two responsive L2CAP ports found, with maximum transmission unit (MTU) size of 4096 and 242 respectively. Flooding of the port with the larger MTU resulted in discoverable mode disrupted intermittently when trying to pair. Calls made had quality issues or were dropped.
Additional observations		
Time discoverable was limited to two minutes, and the user had to enable Bluetooth. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any of the attempted actions beyond pairing. The vehicle recognised a spoofed device as one that has previously paired, although authentication checks failed (probably due to incorrect location on the test laptop of the link key acquired from a previously paired device).		

Table 16: Experimental Results: Vehicle 2 [33]

	Interface characteristics	*Observation*
	Address: xx:xx:xx:C3:4A:64 **Version:** 2.0 **Class:** 0x340408 **Services:** Headset, Sync, HFP, OBEX OPP, UPDATE, Vendor specific SPP1 and SPP2, PBAP (Client), A2DP, AVRCP, OBEX FTP, PANU, Vendor specific Audio Target and Audio Sink **Open ports:** RFCOMM 3, 4, 5, 8, 9, 10, 12 and L2CAP 1, 15, 25, 27	Bluetooth version 2.0 means that vehicle is using legacy pairing exclusively. Vehicle asks for a 4-digit PIN to be entered on testing device, with a default of `0000`. Device class interprets to an audio/video hands-free device This vehicle had five vendor specific profiles, two of which were Serial Port Profiles (SPP). These implies that the system could be manipulated using `AT` commands, although the bespoke aspect might mean that the `AT` commands are also vendor specific. Another interesting point is the Personal Ad-Hoc Network User (PANU) profile, which is used to receive Ethernet packets using the Bluetooth Network Encapsulated Protocol (BNEP).
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	Channel 5 returned `AT+BRSF=63`, other channels returned errors such as `38, Function not implemented`, `103, Software caused connection abort` and `104, Connection reset by peer`. Filesystem was mountable, browsable and writable. There were two directories already present, `recorder` and `update_ftp`. The latter might be connected to the vendor specific UPDATE profile found.
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port scans and flooding caused the vehicle system to make a phone call to "Unknown number". L2CAP ports had very large MTUs, especially 15 (with 1691 bytes) and 27 (with 2048 bytes). The entire headunit refuses to respond thereafter (cannot hang up, physical buttons don't respond and mutes anything that is currently playing). Port flooding also created intermittent `103, Software caused connection abort"` errors.
Additional observations		
User had to enable Bluetooth, but discoverable time was not limited. A small visual alert in secondary screen above the steering wheel is given, but no alert from the main screen during pairing. No alerts were given for all subsequent reconnaissance and data extraction actions. Testing device was added to the paired devices list. The vehicle did not recognise spoofed devices.		

Table 17: Experimental Results: Vehicle 3

	Interface characteristics	*Observation*
	Address: xx:xx:xx:7B:10:69 **Version:** 4.0 **Class:** 0x340408 **Services:** HFP, A2DP, AVRCP, PBAP, MAP MNS, OBEX OPP, Wireless iAP, JCI Reflash Server, Pandora for Android & Blackberry, Stitcher, AHA Radio, Android VDT **Open ports:** RFCOMM 1,2,3,4,5,6,7,8,9,19 and L2CAP 1,23	Bluetooth version 4.0 means that vehicle is capable of using SSP (in this case numeric comparison). Vehicle produces dynamic 6 digit PIN. Pairing with SSP disabled on the test laptop meant `0000` used as PIN. Device class interprets to an audio/video hands-free device This vehicle featured an unusually large number of services. Pandora, Stitcher and AHA Radio were related to music-streaming services. The presence of the Android VDT profile seems to indicate that the vehicle runs Android Auto. Wireless iAP is an Apple proprietary protocol used for data transfer through Bluetooth (there were three ports on the vehicle). Finally, although there was no information regarding "JCI Reflash Server", this service is suggestive of the ability to write to the system.
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	Responds to all `AT` commands sent on all channels with error `12, Cannot allocate memory`, except on channel `2` where the response was error `52, Invalid exchange`. There was no OBEXFTP and therefore no access to the filesystem.
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	L2CAP 1 had an MTU of 246, the other ended with error `111, Connection refused` after sending a packet of size 180 bytes. Subsequent attempted denial of service actions was prevented in all cases, denoted by error `104, Connection reset by peer`.
Additional observations		
Time discoverable was limited to three minutes, and the user had to enable Bluetooth. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any of the attempted actions beyond pairing. The vehicle did not recognise the spoofed device		

Table 18: Experimental Results: Vehicle 4 [33]

	Interface characteristics	*Observation*
	Address: xx:xx:xx:6E:DC:D5 **Version:** 2.0 **Class:** 0x340408 **Services:** Headset, Sync Command, HFP, OBEX OPP, PBAP (Client), Vendor specific SPP1 and SPP2, OBEX FTP, PANU **Open ports:** RFCOMM 2,3,4,5,8,9,10,11 and L2CAP 1,15,23,25,27	Bluetooth version 2.0 means that vehicle is using legacy pairing exclusively. User chooses the PIN. Device class interprets to an audio/video hands-free device Services include Sync Command, which is used to initiate synchronisation from a server, vendor specific serial ports as well as Personal Ad-Hoc Network User, which again indicates the ability to send in Ethernet packets.
	Tests	*Outcome*
Data extraction	`AT` Commands. Filesystem mount and directory traversal	`AT` commands returned `AT+BRSF=63`, `AT+GMI` and `AT+VGS=8`. The latter two deals with displaying the manufacturer for the GSM module and the value of the gain of the speaker respectively. It is unclear whether `+GMI` is empty, or providing an instruction. It is also an indication that there may be cellular capabilities within the vehicle. Filesystem is mountable, browsable and writeable, with two folders `recorder` and `update_ftp`.
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port flooding resulted in headunit dialling ****, and thereafter calls could not be made. Hanging up was not possible, but the rest of the headunit operated normally. Fuzzing on RFCOMM channels resulted in `code=0x43 FORBIDDEN`.
Additional observations		
User had to enable Bluetooth, but discoverability has no time limit. Audio and visual notice was given of pairing attempt, but no alert to successful pairing. Test laptop was not added to the paired list. User is not alerted to any of the attempted data extraction actions beyond pairing. The vehicle recognised a spoofed device as one that has previously paired, although authentication checks failed (probably due to incorrect location of the link key acquired from a previously paired device).		

Table 19: Experimental Results: Vehicle 5

	Interface characteristics	*Observation*
	Address: xx:xx:xx:E5:88:55 **Version:** 2.0 **Class:** 0x340408 **Services:** Audio Sink, AVRCP, PBAP (client), OBEX OPP, HFP **Open ports:** RFCOMM 1 and L2CAP 1,3,23,25	Bluetooth version 2.0 means that vehicle is using legacy pairing exclusively. User chooses the PIN. Device class interprets to an audio/video hands-free device Bluetooth functionality is restricted, with no messaging services, and only standard in-vehicle Bluetooth features enabled
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	`AT` commands returned `AT+BRSF=111` on channel 1. All other channels responded with error `16, Device or resource busy`. No OBEX FTP and therefore no access to filesystem
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port flooding and fuzzing did not yield any observable results.
Additional observations		
User had to enable Bluetooth, discoverability limited to three minutes. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any of the attempted data extraction actions beyond pairing. The vehicle did not recognise a spoofed device as previously paired.		

Table 20: Experimental Results: Vehicle 6

	Interface characteristics	*Observation*
	Address: xx:xx:xx:F0:E7:B8 **Version:** 2.1 **Class:** 0x260408 **Services:** PBAP (client), A2DP, AVRCP, HFP, PANU, "None" **Open ports:** RFCOMM 1,2 and L2CAP 1,23	Bluetooth version 2.1 means that vehicle is using SSP numeric comparison by default, although pairing without SSP was possibly with user choosing the PIN. Device class interprets to an audio/video networking and rendering device Has PANU, which indicates that Ethernet packets can be sent to the system. Curiously, there was also a service labelled "None" assigned to RFCOMM 1 which was open.
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	`AT` commands returned errors `40, Too many levels of symbolic links` and `12, Cannot allocate memory` on all open channels. Filesystem could not be accessed as there was no OBEX FTP
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port flooding and fuzzing did not yield any observable results.
Additional observations		
User had to enable Bluetooth, discoverability time was not limited. Audio and visual notice was given of successful pairing, and test device was added to the paired list, however it added a spoofed name rather than the actual name of the device. User is not alerted to any actions beyond pairing. The vehicle did not recognise a spoofed device as previously paired.		

Table 21: Experimental Results: Vehicle 7 [33]

	Interface characteristics	*Observation*
	Address: xx:xx:xx:8A:81:20 **Version:** 2.0 **Class:** 0x300408 **Services:** PBAP (client), AVRCP, OBEX OPP, SyncML Server, HFP **Open ports:** RFCOMM 1,4 and L2CAP 1,23	Bluetooth version 2.0 means that vehicle is legacy pairing exclusively, with user choosing the PIN. Device class interprets to an audio/video device with object transfer capabilities Has PANU, which indicates that Ethernet packets can be sent to the system. Curiously, there was also a service labelled "None" assigned to RFCOMM 1 which was open.
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	`AT` commands returned error `112, Host is down` on all open channels. Filesystem could not be accessed as there was no OBEX FTP
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port flooding and fuzzing did not yield any observable results.
Additional observations		
User had to enable Bluetooth, discoverability time was not limited. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any actions beyond pairing. Vehicle responded to a SyncML client on test laptop requesting for contact synchronisation in location `text/v-card contacts`, although no information came back, and the session ended (possibly due to an unstable connection). The vehicle recognised spoofed device as a previously paired device, but authentication failed (in this case it was expected as the link key had not been acquired).		

Table 22: Experimental Results: Vehicle 8

	Interface characteristics	*Observation*
	Address: xx:xx:xx:6F:6D:E6 **Version:** 3.0 **Class:** 0x360408 **Services:** OBEX OPP, Audio Sink, AVRCP, HFP, MAP MNS, PBAP (client) **Open ports:** RFCOMM 1,2 and L2CAP 1	Bluetooth version 3.0 means that vehicle uses SSP, in this case numeric comparison, with vehicle generating the six-digit PIN. Device class interprets to an audio/video device with networking, rendering and object transfer capabilities This service has only standard in-vehicle Bluetooth profiles
	Tests	*Outcome*
Data extraction	`AT` Commands Filesystem mount and directory traversal	`AT` commands returned error `12, Cannot allocate memory` and `13, Permission denied` on all open channels. Filesystem could not be accessed as there was no OBEX FTP
Denial of service	Flood L2CAP ports Repeated data push through OBEX channels Fuzz open RFCOMM channels	Port flooding and fuzzing returned errors `12, Cannot allocate memory` and `13, Permission denied`. This may have been a problem with setup (as the "permission denied" error is apparently a common bug [112] in the *Bluez* version used at the time).
Additional observations		
User had to enable Bluetooth, discoverability time was limited to two minutes. Audio and visual notice was given of successful pairing, and test device was added to the paired list. User is not alerted to any actions beyond pairing. The vehicle did not recognise a spoofed device as previously paired.		

Table 23: General summary of test effects on test vehicles

Vehicle No.	*Bluetooth*	*Attack Goal*					
		Data Extraction			*Denial of Service*		
		S	P	F	S	P	F
1	v2.0		x			x	
2	v2.0		x		x		
3	v4.0		x				x
4	v2.0	x				x	
5	v2.0		x				x
6	v2.1		x				x
7	v2.0		x				x
8	v3.0		x				x

S = Success (i.e. positive data which describes the system in some way, or complete component failure), P = Partial success (i.e. only negative data such as errors, or publicly available data, or some part of component service is compromised), F = Fail

6.4 EXPERIMENTAL ANALYSIS

This section explores some of the apparent themes and common threads from the experimental results on the eight vehicles tested. Implications of these themes are also discussed.

6.4.1 *Characteristics*

There were similarities between implementations of Bluetooth (based on the combination and nature of services offered), not between vehicles of the same OEM, but between vehicles who had the same Tier 1 supplier for the headunit. The converse, where vehicular implementations used different Tier 1 suppliers but were manufactured by the same OEM having no similarities, was also true.

For example, vehicles 2 and 4 had different manufacturers, but the same Tier 1 supplier (see Table 14) and clearly displayed the same sets of vendor specific serial port profiles, as well as the PANU profile. They even reacted in a similar way to port flooding (see Section 6.4.3). Likewise, vehicles 5 and 8, which also share a Tier 1 manufacturer but not an OEM also shared similar characteristics. However, without

vendor specific profiles, this may be less straightforward to compare without system specifications or information.

Conversely, vehicles 6 and 7 (being from the same OEM) had different Tier 1 suppliers. Vehicle 6 had a PANU profile, and a nondescript "None" service, whilst vehicle 7 had a SyncML Server profile. Both returned different reactions to `AT` commands.

Although no statistics are available, there are indications here that profiling or reconnaissance on Bluetooth implementations in automotive headunits should be based on the Tier 1 supplier rather than the manufacturer of the vehicle.

6.4.2 *Pairing and connection*

The pairing process differed (at least mechanically) on the vehicles tested, with some generating PINs, others with hard-coded PINs, and still others asking the users to select the PIN.

Five (vehicles 1, 2, 4, 5 and 7) out of the eight vehicles tested used the legacy pairing mechanism exclusively. This is the more insecure of the two pairing mechanisms specified by Bluetooth SIG (see Chapter 3), and could be considered a design weakness. A passkey was always required, although implementations differed in that some generated a PIN, whilst in other vehicles, the user was required to choose the PIN. In one instance, the PIN was set to '0000' by default (vehicle 2), although this was user customisable. The issue with the latter, however, is that it is easily guessable and increases the risk of an unauthorised connection, especially if the user did not change it.

Where the vehicles used SSP (where we observed that the mode most in use is numeric comparison), disabling SSP on the test laptop meant that pairing occurred with the legacy pairing mechanism (a form of downgrading attack [9]). Here, too, there was an occurrence of the default PIN being '0000' (vehicle 3). The other two required the user to choose the PIN. The implication of this is that even where the newer SSP pairing mechanism is in use, bypassing such measures could be straightforward because of the weakness of the default PIN.

The number of open ports is usually dependent on whether a user is paired or connected to the vehicle. However, the OBEX OPP profile remained open in vehicles 1 and 2. Although data extraction through this port was unsuccessful, techniques such as repeated pushing of files through this port could help fulfil alternative attack goals (such

as denial of service). The latter in this case was not performed due to the invasiveness of the test and the risk of damage to the vehicle.

Once pairing and connection was established, all vehicles universally reconnected with the test laptop as soon as it came into range (which could, of course, be extended). The ramification of this is that an attacker would only need to go through (or compromise) the pairing process once.

The window in which a vehicle remained discoverable also varied. Half the vehicles tested (vehicle numbers 1, 3, 5 and 8) had a time limit of either two or three minutes where the discoverability of the interface was enabled. The other half held the discoverability window open indefinitely, which may leave the vehicle open to opportunistic adversaries. However, this is mitigated somewhat by the fact that in all cases, the user had to enable Bluetooth discoverability which offers some protection from the risk of exploitation. Note that it is only a mitigation (rather than a true defence) as brute-force scanning (walking through the lower address part (LAP) of the Bluetooth address and incrementing by one before sending a query [66]) is possible, and a device *will* respond to an inquiry if queried directly. Feasibility of this attack depends on having enumerated some of the address bytes (for example, through knowing the address bytes of the OUI).

Vehicles 1, 4 and 7 recognised a spoofed device as a previously paired device and automatically tried to reconnect as soon as the spoofed device was within range. In two cases (vehicle 1 and 4), authentication failed. Although the correct link key (which is calculated during the pairing phase and used for all transactions thereafter - see Chapter 3) was acquired, the location it was placed in the test laptop might have caused the rejection of authentication. The latter is likely a setup issue. Since Bluez (the Bluetooth stack on Linux) had no documentation regarding the manipulation of link keys, the process was (and in the future will continue to be) trial and error. In vehicle 7, authentication failure was expected since at the time we had not yet acquired the correct link key.

6.4.3 *Implementation weaknesses*

An interesting finding was the ability to mount a filesystem with read and partial write access; through this entry point, any number of crafted applications could be placed on to the vehicle to dis-

rupt operations. There were directories already present (`recorder` and `update_ftp` on vehicles 2 and 4 (where both vehicles, incidentally, also had the same Tier 1 supplier). The latter might be connected to the `UPDATE` service profile found on vehicle 2. Although write access was verified by creating directories, there was no attempt to place malicious files or to fuzz (for example by using directory names containing non-standard characters). This would have revealed vulnerabilities that stem from this design weakness. However, the owner did not give consent for this. Further experimentation along these lines could reveal more about this feature.

There were also vendor specific profiles found on three of the vehicles. Vehicles 2 and 4 contained what appeared to be serial port profiles. `AT` commands elicited Bluetooth error `104, Connection reset by peer`, although its functionality was not probed further. Future work may include traffic sniffing and analysis during normal course of operations which might yield more information. The vendor specific profiles on vehicle 3 indicated profiles that allowed for various streaming music services. Experimentation through Apple's "Wireless iAP" synchronisation service could also reveal more (although due to resource constraints, this was not tested at the time). There was a particular profile named "JCI Reflash Server". Although there is no publicly available information, the name is suggestive of a feature that could be exploited. Further exploration would require sending in information and seeing returned data through use of a traffic sniffer or analyser.

Since hands-free phone calls are one of the more prominent features of Bluetooth in vehicles, it was unsurprising to see the Phone Book Access Profile (PBAP) on some of the vehicles. This allows phonebooks to be synchronised from a mobile phone (or other device holding a phonebook in the correct format) to the vehicle. The 'client' status denotes that this only goes one way, from the remote device to the vehicle. Tools such as *nOBEX* could be used to fuzz this particular feature by uploading contacts or phone numbers that have non-standard characters, or are past a certain length. However, there was reasonable expectation from experimental analyses performed on vehicles that this might damage the software on the headunit in some form, so the latter was not tested.

Another feature of interest was that a synchronisation service profile (SyncML Server, Sync, Sync Command) was present in many of

these vehicles. These synchronisation profiles are generally used to synchronise phonebooks and other personal information such as contacts and calendars between phone and vehicle. Although there was no personal data extracted, in at least one instance, a connection (albeit unstable) with an alternate SyncML client was established (Bluez has no such support); the setup could be revised to try and correct for this (for example by asking for contacts or other data from different locations) and verify whether any data could be extracted through this method.

The presence of the Personal Ad-Hoc Network User (PANU) profile on two of the vehicles also bears some scrutiny. This service is able to transfer Ethernet packets across a connection through the use of the Bluetooth Network Encapsulation Protocol (BNEP). There are three security modes used by this profile. The first is "non-secure", where a device does not initiate any security procedures. The second is service-level enforced security, where security procedures are not initiated before a channel is established at L2CAP level. Lastly, the link-level enforced security mode initiates security procedures before the link set-up at the LMP layer [18]. LMP controls the radio link between two devices. The mode used by the PANU profile in this case is so far unenumerated, but represents a potential alternative method to send in (Ethernet) packets that could compromise the headunit.

The denial of service attacks, particularly the port flooding ended with vehicles 2 and 4 dialling either an "Unknown number" or "****" respectively. These are implementation errors which have caused a vulnerability to port scanning. In the case of vehicle 2, the entire headunit refused to respond thereafter. This included the media and CD player, radio, virtual buttons or physical buttons. This behaviour also continued for three minutes after the entire vehicle was switched off and locked. The headunit remained non-operational until the test laptop disconnected the Bluetooth connection. Although the process required pairing, some premeditation regarding pre-pairing (or using an already paired device) could cause such behaviour, rendering a denial of any information or entertainment service so long as the vehicle stays within range. In the case of Vehicle 4, although there was similar behaviour in dialling "****", the infotainment system remained usable; menus could still be accessed and button presses worked as normal. However, none of the Bluetooth functionality was accessible (no calls could be made, and contacts could not be accessed). In the con-

text of a real-world attack scenario, an example could be that such an attack could be used to interrupt or stop a driver from accessing emergency calling services. Vehicle 1 showed no reaction to port flooding attacks in the normal course of operations. However, calls made by the headunit through the Bluetooth connection were dropped and call quality suffered during these attacks, especially with ports with large MTU sizes (above the default 672 bytes). This result was reproducible but was not present in every test run. In the real-world context, this could be used to interrupt calls that are made (for example, to emergency services). This result could not be replicated in other vehicles, although this could be explained by the fact that this vehicle had the largest MTU (4096 bytes) of any of the vehicles tested, and therefore flooding might have been more effective in this case.

6.4.4 *Covert Actions*

It has been posited that user interaction within procedures can improve security [164], although there may be conflicts with regards to safety, in terms of what feedback is given to the driver of the car, especially during drive time.

The first general observation, however, is that, beyond the pairing process, the human is not usually alerted nor in the loop when other actions were performed. Thus, `AT` commands, filesystem mounting or port floods generally went unacknowledged at the graphical front end. The exception to these were where port flooding caused the headunit to dial unknown numbers (vehicles 2 and 4), although these were not alerts, but rather alerting events. The implication is that attacks could be attempted on the vehicle without the driver being any the wiser, and the results thereof (such as data exfiltration) could likewise be covert. All are weaknesses in implementation. Mitigation could be provided through a situational awareness display, or through knowledge of what protections are available. Although this is still conceptual, the mandatory use of a "cyber dashboard" has already been proposed by impending legislation such as the US SPY Car Act 2017 [1].

There was an instance of a vehicle which did not add the test laptop to the paired devices list (Vehicle 4), despite the fact that there was a successful pairing and an active connection. Reconnection to the system under test with different device classes were attempted.

Most smartphones, for example have a class device of `0x5a020c`, and spoofing this from a laptop caused several vehicles to recognise the laptop as a smartphone (rather than as a media player which was usually the case). Whether this changes the behaviour of the vehicles or affects whether it is placed on the paired devices list could not be ascertained. This presents an increased risk, as not only covert attacks would be possible, but there would be no record (at least on the surface) that would point to an unauthorised device having paired or connected.

6.5 CONSTRAINTS

Discussed in this section are the constraints on the experiments performed.

6.5.1 *Commercial Confidentiality*

Organisations such as Information Sharing and Analysis Centres (ISAC) have been around since 1999 (in the US) [114], but the addition of an automotive specific centre has only been a recent development [8]. However, even with the publication of best practices, and with OEM involvement, the centre still emphasises anonymity regarding information that is shared and disseminated [8]. This is telling of the current nature of the automotive industry. Because of this, technical specifications, source code, binary files and other information that might be helpful for a white-box investigation are unlikely to be available. This impact is felt on all experiments performed in this thesis, and is the primary reason for taking a black box approach.

6.5.2 *Risks to Vehicles*

There are areas of further interest beyond the work that was done here. Many of the tests designed could not be performed as the potential integrity of the vehicles (and thereby its safety) was a concern. Assurances were given to owners of the vehicles that none of the tests would be excessively invasive. This meant that, by necessity, many tests (primarily involving fuzzing) were not performed.

There were also many services that could be explored further. This includes the IrMC Sync command (via Infrared), the Personal Ad-Hoc Network User Service (which allows for Ethernet packets to be sent via encapsulated Bluetooth packets), as well as PBAP synchronisation (dealing with how the phone book is synchronised between devices). Several other tools have since been created since the inception of this method. One of the most relevant is *nOBEX* [113], which allows the manual fuzzing of the automotive headunit via manipulation of contacts and calendar items. However, uploading malformed data may have caused the headunit to malfunction (as has happened in studies such as that undertaken by Checkoway, McCoy, Kantor, *et al.* [36] and Miller and Valasek [110]).

One of the vehicles appeared potentially susceptible to spoofing attacks, in that it recognised an illegitimate device with the same Bluetooth address as a previously paired device. Authentication did not proceed (as discussed in Section 6.4.2). However, the location of the link key on the vehicle (assuming access to a previously paired phone is restricted) can be derived from where it is usually stored on conventional Windows or Linux installations (assuming the vehicle is using Microsoft Windows Auto Embedded or QNX). This subject requires more exploration, which at this point in time was not in scope as it would require invasive investigation involving physical removal and bit-by-bit imaging of the automotive headunit.

An implication of the work here is that privacy could be compromised. However, there were no explicit circumstances in the vehicles that were tested where privacy was affected (although an attack tree could be created with the attack goal of privacy violation as a direction for future development of tests). Thus, other tests could include tracking the address, name and class of device, along with the mapping the signal (RSSI) strength to a variety of physical distances. This would also feed into the range extension attacks, as the initial distance (usually 10 metres in a class 2 device) could affect the extent to which such devices would remain within range. This would have required legal advice regarding both the use of antennas and the co-operation and knowledge of the drivers. At this point in time, such an exercise was deemed out of scope for this thesis.

There were other potential vehicles that were used for the threat intelligence study (Chapter 3.4) which could have been investigated further. However, owners did not always give permission to form an

active connection (or send data) to the vehicle. Because of the expense of these systems, the number and nature of vehicles that we could do even truncated tests on was limited.

6.5.3 *Limited Number of Vehicles*

Vehicles are expensive, and because the ones of most interest are the vehicles containing sophisticated software, this expense is amplified. Facilities are also needed to house these cars, and because of the expense, is also likely to be a shared resource.

There is also a safety concern with regards to the vehicles tested. Experimentation on both the headunit and through aftermarket devices (as described in the next chapter) caused intermittent issues, such as constant battery drainage which led to battery damage, active and persistent diagnostic problem indicators and slowdown of the headunit software.

The above becomes an especial issue where the vehicle is privately owned. Because of these risks, the number of vehicles available were limited. To that end, eight cars is a small number to test, but the findings still give a useful baseline insight into what might be implemented on the vehicle.

Work is underway to create testbeds [53] using industry standard tools that could mitigate these risks, but still give a trustworthy platform on which to test and analyse attacks. However, these are still nascent, and much of the information required to accurately emulate a vehicle remains publicly unavailable.

6.6 SECURITY ASSURANCE

A security assurance case necessarily requires that a claim (such as "a system is acceptably secure") should be accompanied by evidence. The framework and the results from the application thereof would form part of this case.

There are three aspects of the framework described in previous sections that could be used to support a security assurance case:

- Firstly, a baseline state could be obtained with what is currently in production, reflecting realistically what an adversary who is not an insider might find;

- Secondly, a similar test could be run iteratively (and indeed, the tool is designed to be run many times) and the results could be compared after countermeasures and controls are put in place. This would result in a differentiation in ratings that could serve as evidence of improvement;
- Thirdly, many of the tests were not run to completion because of the constraints and limitations based on lack of technical information as well as risks to test vehicles. To mitigate this and give guidance as to the test cases that could be used should circumstances allow, potential ratings could also be given based on the worst case scenario; and
- Finally, the severity classification scheme as described in Chapter 5.1.4 could be used as a relative measure to ascertain which risks should be addressed before others (i.e. evidence for prioritisation). This is discussed further below.

6.6.1 *Case Study*

The severity classification method (and its validation) was, by necessity, created later in the study. As such, seven of the eight vehicles previously tested on for data extraction and denial of service were no longer available for use. The real world aspect of this study is thus limited to the single test vehicle available.

6.6.1.1 *Severity classification: data gathering of manual observations*

Each rating is given as $S_p i$, $S_o i$ where S_p represents the privacy severity rating, and S_o the operational severity rating and where $i \in \{0-4\}$.

Once the test runs were completed, questions were asked of the tester with regards to:

- Service profiles and its nature:
  - Were there profiles that are named suggestively?
  - Were there vendor specific profiles?
  - Were there synchronisation profiles?
  - Were there Personal Ad-Hoc Network (PAN) profiles?

  Certain service profiles mean that personal information can be synchronised between two devices, which might impact pri-

vacy. Others may offer more access to the system itself, either by broadening the attack surface (such as by allowing Ethernet packets through e.g. through the PAN profile) or bespoke services which may have implementation flaws. Suggestive profiles (such as "Reflash Server") may help narrow the scope or provide a target for an adversary. These affect the privacy rating and potentially the operational rating.

- PIN behaviour:
  - Was the PIN dynamic?
  - Was the PIN customisable by user?
  - Was the PIN easily guessable?

  Each of these questions (depending on a positive or negative answer) would affect the risk of an adversary being able to compromise the Bluetooth connection through, for example, eavesdropping. In the worst case scenario of a static, fixed and easily-guessable PIN, the risk would be far greater than if the PIN had been dynamic or customisable. This would affect both operational and privacy ratings.

- Data returned by the vehicle:
  - Was there information about the vehicle returned?
  - Was there information about the user of the vehicle returned?

  These questions would affect the privacy ratings given, with the latter being the highest severity.

- The behaviour of the vehicle during testing:
  - Was there discernible operational impact on the system during testing?
  - Was there a reaction on the user interface?

  These questions were both used to discern the impact on the system, and whether any alerting effects resulted. This affects the operational rating during classification.

Other aspects such as vehicle tracking through the Bluetooth address (which affects privacy), and whether there were open ports (which potentially affects operations) are pulled from the logs created

on the findings and classified automatically. Thus, the combination of answers used in conjunction with the findings of the tests resulted in ratings given to each aspect of the test.

This particular part of the tool also checks through the tree to give automatic ratings. For example, where there is no OBEX FTP, the mounting and traversal attacks were not performed. This would automatically result in an S_p0 and S_o0 rating.

Recall that only the privacy and operational aspects of the EVITA classification scheme were considered, since safety (and safety analysis) is considered out of scope for this thesis. Financial ratings were also considered out of scope as mechanisms for financial transactions on vehicles are not yet widely deployed, and exploring financial loss from vehicle theft would also require information that is not available.

6.6.2 *Assignment of Severity Classifications*

The severity ratings that were assigned for the data extraction and denial of service attack goals are given in Figure 11 and Figure 12 respectively.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	Address: xx:xx:xx:34:8A:2D	Sp2,So0	Address unique to each device
POT		Sp3,So0	Vehicle Tracking possible
ACT	OUI: xxxxxxxxxxxxxxxxxxxxxxxx	Sp1,So0	Anonymous data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	OS: NULLOS	Sp0,So0	No data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	Generic services (no suggestive)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Generic services (no bespoke)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Service port no.: 7	Sp1,So1	Sync services found
POT		Sp3,So3	Unauthorised sync --> Sp3, Fuzzing --> So3
ACT	Generic services found (no PAN)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Pairing: legacy	Sp1,So1	Legacy pairing found,
			susceptible to MITM, deprecated spec
POT		Sp3,So2	Attack goal/services offered dependent
			--> data loss, MITM or DoS
ACT	Open RFCOMM ports: [1, 4]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Open L2CAP ports: [1, 3, 23, 25]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	AT command: System data acquired	Sp1,So1	Data extract not discernible,
			anonymous data acquired
POT		Sp3,So3	Potentially more data could
			be acquired via other AT commands
ACT	OBEXFTP: MountFailed or MountNA	Sp0,So0	Mount failed, or OBEXFTP not found
POT		Sp0,So0	
ACT	OBEXFTP: Directory Traversal failed	Sp0,So0	Traversal failed, or OBEXFTP not found
POT		Sp0,So0	

Figure 11: Severity classification ratings for the data extraction test suite

The address, as a unique identifying factor of the headunit, could also be used to track a vehicle. In terms of the OUI, the fact that this organisation is known could then lead to further reconnaissance (including research into known bugs or software defects), which could lead to significant impairment. The operating system could not be enumerated by the tool in this case, and so an automatic rating of o

on both privacy and operational fronts was assigned. This could be adjusted based on the results of manual observation.

The services were all generic (no suggestive or bespoke services), with no PAN services identified. Again, this is all anonymous data (S_p1), but operationally has no impact. Legacy pairing was identified, and so given an actual operational rating of S_o1, since the impact is indiscernible, but still presents a weakness. Potentially of course, the rating is much higher since compromising the pairing could lead to any number of attacks (including against privacy).

The presence of open ports meant that an actual operational rating of S_o1 was assigned, since this allowed us to acquire system information. However, operationally, the right `AT` commands or number of packets to send could be enumerated and this may result in a higher potential rating. Since there was no OBEX FTP service on the vehicle, both the mounting and directory traversal attacks were not carried out, and therefore both were automatically assigned S_p0, S_o0 on all fronts.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	Address: xx:xx:xx:34:8A:2D	Sp2,So0	Address unique to each device
POT		Sp3,So0	Vehicle Tracking possible
ACT	OUI: xxxxxxxxxxxxxxxxxxxxxxxx	Sp1,So0	Anonymous data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	OS: NULLOS	Sp0,So0	No data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	Generic services (no suggestive)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Generic services (no bespoke)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Service port no.: 7	Sp1,So1	Sync services found
POT		Sp3,So3	Unauthorised sync --> Sp3, Fuzzing --> So3
ACT	Generic services found (no PAN)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Pairing: legacy	Sp1,So1	Legacy pairing found,
			susceptible to MITM, deprecated spec
POT		Sp3,So2	Attack goal/services offered dependent
			--> data loss, MITM or DoS
ACT	Open RFCOMM ports: [1, 4]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Open L2CAP ports: [1, 3, 23, 25]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Flooding: indiscernible impact	Sp1,So0	System information, no impact
POT		Sp1,So3	More testing could cause further DoS
ACT	ObexStress: no impact,	Sp1,So0	System information acquired, no impact
	system info acquired		
POT		Sp1,So3	Further testing could cause further DoS

Figure 12: Severity classification ratings for the denial of service suite

The first sets of results from the denial of service tests in Figure 12 are identical to the data extraction results since the reconnaissance aspects were performed in the same vehicle. They are included in the figure for clarity. For the purposes of the denial of service tests, only the last two sets of results are discussed.

Flooding caused no discernible impact, however, we were able to enumerate the maximum transmission units of the open L2CAP ports on the vehicle, therefore system information was available. This re-

sulted in the assignment of S_p1 (for anonymous data acquired), but since there was no discernible impact, the operational rating was set at S_o0. Potentially, however, further testing (with more specific unit sizes, or with more packets or over a longer period of time) could cause a denial of service in any number of Bluetooth functionalities in the worst case scenario.

Likewise, stressing open RFCOMM ports caused no impact, although again system information was required (even if it was the fact that the action was forbidden) and therefore given the same rating as above. Potentially, finding the right combination and length of malformed data could also cause denial of service, and the ratings adjusted accordingly.

Having no information regarding internal paths or interfaces precludes us from formal verification and validation. There are also no set of expected outputs in the given context of automotive systems, since vehicular experimental analysis has typically concentrated on other technologies (discussed in Chapter 2.1.1). Thus, this severity classification was validated by two domain experts (biographies are available in Appendix A.2).

6.6.3 *Theoretical Ratings*

Although the practical testing to semi-automatically assign ratings could only take place on one vehicle, information (both manual observations and logs) are available such that we can form theoretical classifications by manual means. The ratings follow the same classifications as described in the SAE J3061 standard (see Table 1 in Chapter 2.2.1).

An example of the theoretical rating is given in Figures 13 (for the data extraction attack goal) and 14 (for the denial of service attack goal).

The Bluetooth address is unique to each headunit and by extension the vehicle. Such an ability to identify a vehicle justifies an S_p2 rating. Operationally, the severity was set at S_o0 as there was no operational impact. Since identification is possible, tracking is also possible. Hence, the potential severity of S_p3 was assigned.

There were no suggestively named or bespoke services, however synchronisation services were found. This in itself is anonymous data which had no operational impact discernible when probed which led

to the S_p1,S_o1 ratings. Potentially, this could lead to unauthorised synchronisation or fuzzing, which resulted in the POT S_p3,S_o3 rating. Legacy pairing likewise is anonymous data with no operational impact discernible (S_p1,S_o1). Compromising the pairing mechanisms could lead to driver or vehicle tracking, and could potentially cause a performance degrade (in the head unit) and so a rating of S_p3,S_o2 was assigned.

The ability to elicit system information using AT commands as well as being able to mount the filesystem both led to an ACT rating of S_p1,S_o1, but could potentially affect both privacy and performance. AT commands could be used to extract phone numbers, and mounting with read/write access could mean the injection of malicious files. This led to the POT rating for these aspects at S_p3,S_o3.

Open ports are also a danger, as flooding these ports led to the headunit being hung on a call, with none of the virtual or physical buttons responding (hence the S_o3 rating).

Although the assignation was completely manual in these instances, the exercise serves as a useful point of comparison to the practical example carried out above.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	Address: xx:xx:xx:C3:4A:64	Sp2,So0	Address unique to each device
POT		Sp3,So0	Vehicle Tracking possible
ACT	OUI: xxxxxxxxxxxxxxxxxxxx	Sp1,So0	Anonymous data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	OS: NULLOS	Sp0,So0	No data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	Generic services (no suggestive)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Vendor specific profiles found	Sp1,So0	Anonymous data acquired
POT		Sp3,So3	Fuzzing --> So3,
			personal data could be acquired
ACT	Service port no.: 4	Sp1,So1	Sync services found
POT		Sp3,So3	Unauthorised sync --> Sp3, Fuzzing --> So3
ACT	PAN service found	Sp1,So0	Anonymous data acquired
POT		Sp3,So3	Ethernet and TCP/IP attacks possible
			(i.e. larger attack surface)
ACT	Pairing: legacy	Sp1,So1	Legacy pairing found,
			susceptible to MITM, deprecated spec
POT		Sp3,So2	Attack goal/services offered dependent
			--> data loss, MITM or DoS
ACT	Open RFCOMM ports: [3-5,8-10,12]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Open L2CAP ports: [1,15,25,27]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	AT command: System related data	Sp1,So1	Data extract not discernible,
			anonymous data acquired
POT		Sp3,So3	Potentially more data could be
			acquired via other AT commands
ACT	OBEXFTP: Successful mount	Sp1,So1	Mount success, indiscernible impact
POT		Sp3,So3	Read/write access, potential compromise
ACT	OBEXFTP: Directory Traversal	Sp0,So0	Traversal failed, or OBEXFTP not found
	failed		
POT		Sp0,So0	

Figure 13: Theoretical classification of vehicle 2 for data extraction

As with the case study above, the ratings vary between S0 and S3 as we are testing individual headunits rather than anything that could affect multiple vehicles. In this case, it becomes quite easy to pick out that a successful mount could lead to compromise of the

No.	Attribute	Rating (S0-S4)	Evidence
ACT	Address: xx:xx:xx:C3:4A:64	Sp2,So0	Address unique to each device
POT		Sp3,So0	Vehicle Tracking possible
ACT	OUI: xxxxxxxxxxxxxxxxxxxx	Sp1,So0	Anonymous data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	OS: NULLOS	Sp0,So0	No data acquired
POT		Sp1,So3	Further recon may lead to So3
ACT	Generic services (no suggestive)	Sp1,So0	Anonymous data acquired
POT		Sp1,So0	
ACT	Vendor specific profiles found	Sp1,So0	Anonymous data acquired
POT		Sp3,So3	Fuzzing --> So3, personal data could be acquired
ACT	Service port no.: 4	Sp1,So1	Sync services found
POT		Sp3,So3	Unauthorised sync --> Sp3, Fuzzing --> So3
ACT	PAN service found	Sp1,So0	Anonymous data acquired
POT		Sp3,So3	Ethernet and TCP/IP attacks possible (i.e. larger attack surface)
ACT	Pairing: legacy	Sp1,So1	Legacy pairing found, susceptible to MITM, deprecated spec
POT		Sp3,So2	Attack goal/services offered dependent --> data loss, MITM or DoS
ACT	Open RFCOMM ports: [3-5,8-10,12]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Open L2CAP ports: [1,15,25,27]	Sp1,So1	Port scan invisible, anonymous data acquired
POT		Sp3,So3	Open ports --> Sp3, port scan --> DoS, So3
ACT	Flooding: discernible impact	Sp1,So2	System information, headunit hung on call
POT		Sp1,So3	More testing could cause further DoS
ACT	ObexStress: no impact, system info acquired	Sp1,So0	System information acquired, no impact
POT		Sp1,So3	Further testing could cause further DoS

Figure 14: Theoretical classification of vehicle 2 for denial of service

headunit and that the availability of vendor specific profiles and the PAN profile enlarges the number of attacks or techniques that could compromise the system.

Similarly, Figure 14 shows that privacy is less of an issue (from the perspective of only trying to cause a denial of service as per the attack goal), but that there was definitely discernible impact by flooding the open ports.

Compared to the classification of the first vehicle, this vehicle seems on the surface to be more insecure; the vehicle presented more effects, and there are more avenues for further investigation that could lead to a more severe compromise. However, it should be noted that the vehicle from which the theoretical results were put together is significantly more advanced, with richer feature sets. This should be taken into consideration if direct comparison between the systems-under-test is required.

6.7 DISCUSSION

As can be seen from the results of the classification, there are some aspects that can be used almost immediately.

Anything that is classified as S_p0,S_o0 can be added to the security assurance case as low risk, and therefore low priority. Conversely, anything with a classification of $S4$ in any aspect can be targeted for the development of countermeasures since this rating indicates a

risk to multiple vehicles. When this process is considered complete, comparisons can be drawn with the initial rating, and if considered acceptably addressed, can also be added to the security assurance case as evidence of risk analysis and risk reduction.

Everything else in the middle would most likely depend on the components being targeted. These tests are performed on the head-unit where personal data is most likely to be stored, so privacy might potentially be given more precedence. For example, given a choice between S_p1,S_o2 and S_p2,S_o1, the latter might be the more likely candidate to target for improvement. This may seem intuitive given only two parameters, but in the event where all four aspects of the EVITA classification scheme are in play, this could aid in the decision making process.

Once these classifications take place, they could be used as evidence in security assurance cases not dissimilar to the Automotive Security Assurance Levels (ASEALs) as proposed by studies such as Bayer, Enderle, Oka, *et al.* [10]. Placing it in such a structured manner could also help scope the breadth and depth of tests to be performed, in addition to the priority of the testers or the owners of the system-under-test.

Finally, the rationale given for the worst-case scenario (or the potential ratings) is intended to be a guideline to the starting point for new test cases based on the information acquired during the reconnaissance phase. These also provide a feedback pathway to threat analysis and risk assessment as required by J3061 during the concept development phase. Alternatively, these can be used as guidance to what might be performed by a malicious adversary to complete the attack, since many of the tests pull up just short of an invasive attack due to the risk to test vehicles (see Section 6.5.2).

6.8 CONCLUSION

It is not inconceivable that each of the weaknesses found here could lead to a specific exploit. For example, being able to mount and browse the filesystem with read-write abilities could lead to any number of buffer overflow, fuzzing or flooding attacks to the detriment of safety. Being able to bypass secure simple pairing with a default easily guessable PIN would mean that the risk of unauthorised access to the system is increased, with covert actions taken on the vehicle

demonstrated to be possible. Freezing the headunit, or being able to flood ports (some of which are open even without pairing) could mean that vital services such as emergency calls (i.e. "eCall" services, mandated by the EU and to be deployed on all vehicles by 2018 [50]) could either be dropped or not get through at all.

Once testing is complete, the severity ratings that are assigned can be used to prioritise development of countermeasures, to add evidence to a security assurance case and the rationale behind the worst-case scenario ratings could be used as guidance for further tests.

7

EXPERIMENTAL APPLICATION: AFTERMARKET DEVICES

Highlighted in this chapter is a case study and experiments illustrating the potential threat that an aftermarket device connected to the Onboard Diagnostics (OBD-II) port may pose to the vehicle as a whole.

An overview of the context as well as the vehicular communications technology in use here is given in Sections 7.1 and 7.2 respectively, followed by the objectives of the experimental analysis in Section 7.3.

Experimental setup, results and analyses are presented in Section 7.4 followed by an exploration of what these results might mean in a security assurance context in Section 7.5. Finally, discussions and conclusions are given in Sections 7.6 and 7.7.

7.1 OVERVIEW

The OBD port is used to provide diagnostic data for regulatory officials, Original Equipment Manufacturers (OEMs), mechanics and consumers. Initial experimental analyses (such as that performed by Koscher, Czeskis, Roesner, *et al.* [93]) used a wired connection. However there has been a proliferation of wireless aftermarket devices which attach to this port, some of which have been used in illegal activities such as programming a new key in order to steal the car [29].

The devices being used to test the vehicle (also known as dongles) connect to the OBD port for communication with vehicular systems from external sources. This is a method popular with black-box insurance telematics devices (whereby insurance premiums are reduced in return for "good" driving, measured by this device using "driving style"). The determination of good driving style is usually established through usage patterns, typically drawing from parameters such as acceleration, cornering speed and harshness of braking [67]. Market

forecasts predict that up to 60% of vehicles in the UK will have some kind of insurance telematics attached by 2020 [67].

Vehicle data that is available through the OBD port is neither encrypted nor typically access controlled, either physically or digitally. Beyond the legal requirement for the OBD port, manufacturers also use it for maintenance, diagnostic tests or other manufacturer specific purposes.

7.2 COMMUNICATIONS

Typically, messages that are sent into the OBD port are either raw Controller Area Network (CAN) or diagnostic messages. These two areas of intra-vehicular communications are introduced below.

7.2.1 *CAN Messages*

The CAN protocol is the primary mode of communication inside the vehicle. The latest version is CAN 2.0, first specified in 1991 [129] and embodied as an ISO standard (ISO11898) in 2003. These CAN messages carry much of the information needed for the operation and control of the vehicle.

The standard CAN packet comprises (up to) 11 bits for the message ID, followed by (up to) 8 bytes of data, then a cyclic redundancy check (16 bits) for error detection. The extended CAN frame format uses 29 bits instead for the message ID with slightly different configurations of bits to allow for this. We concentrate here on the standard CAN message only. The full standard CAN frame format can be found in Table 24, with descriptions of each set of bits in Table 25. The full 8 bytes of data need not be used. Information for a door sensor, for example, may only require 1 bit. Conversely a message can be spread across many frames, with various data lengths and offsets.

1-bit	11-bit	1-bit	1-bit	1-bit	1-bit	4-bit	$\leqslant$ 64-bit	16-bit	2-bit	7-bit	7-bit
SOF	IDE	RTR	IDE	r1	r0	DLC	Data	CRC	ACK	EOF	IFS

Table 24: CAN frame format [129]

Arbitration, should nodes on the CAN network transmit simultaneously, is based on message prioritisation. This prioritisation is determined using the message ID, with the lowest ID being the high-

est priority. That being the case, implementation usually means that mission-critical messages are the ones assigned lower IDs.

Assignment of IDs along with data payload is manufacturer specific, however, reuse is common to save on the cost of redesigning a network [125]. Although CAN data is not typically encrypted, reverse engineering can be a difficult process considering the volume and variety of content that is transmitted. This is especially the case without a CAN database, which contains definitions for every message and signal. This file is often highly confidential. However, specific CAN messages for discrete events (such as unlocking doors) can be obtained in a relatively straightforward manner through trial and error experiments.

CAN data is transmitted on a CAN network in a bus configuration. Therefore any Electronic Control Unit (ECU) on the network has access to all messages. There is no addressing; instead each ECU is programmed to listen to a set of CAN IDs, which triggers some pre-determined functionality.

7.2.2 *Configuration and Diagnostic Messages*

Parameter IDs (PIDs) are used to perform diagnostic functions or request data from the vehicle specifically through the OBD-II port. This is done through a query-response mechanism, where a PID query comprises the CAN ID `7DF`, followed by a number of (typically three) data bytes. The first byte is the data length (usually 02) with the second byte being the *mode* and the third byte typically being the PID. The combination of modes and PIDs can then be sent into the CAN bus, and a response should be received from whatever in-vehicle module is responsible (if any). The response CAN ID is typically 8 (in hex) higher than the message ID that the responding ECU answers to. A response of `NO DATA` usually indicates that the vehicle has not returned anything, a response beginning with `7F` in byte 2 means that the vehicle does not recognise the request (with qualifying factors). For example, the vehicle might return `7F` if the requested data could not be obtained due to a malfunctioning sensor.

The first ten modes (`01` to `0A`, described in SAE J1979 (E/E Diagnostic Test Modes) [134]), are standard and generic to all compliant vehicles. In these standard modes, the PID is only the 2nd byte, with the 3rd to 8th byte unused. With non-standard modes, the PIDs could ex-

Acronym	*Description*
SOF	Start of File
IDE	The Identifier Extension establishes the priority of the message. The lower the binary value of the ID, the higher its priority. A CAN message frame with an 11-bit ID is a *standard* frame. One with a 29-bit ID is an *extended* frame.
RTR	Remote Transmission Request; if this bit is dominant (i.e. 0), more information is necessary from another node
r0/r1	Reserved bits originally, but now in use in some implementations to identify XOR masked CAN messages
DLC	The Data Length Code contains the number of data bytes to be transmitted
CRC	Cyclic Redundancy Check for error detection
ACK	The Acknowledge bit is overwritten (from recessive (1) to dominant (0)) to acknowledge validity
EOF	End of File
IFS	The interframe space contains time required to move a received frame to the message buffer area

Table 25: CAN frame format descriptions [129]

tend to the 3rd byte. Manufacturers are not obliged to implement all standard commands and additionally could also define functions for non-standard PIDs. There is much information that could be gathered using PIDs to interrogate the vehicle. For example, sending the mode `09` with PID `02` retrieves the Vehicle Identification Number (VIN). The VIN is unique to the vehicle and is used for many activities, from vehicle maintenance to recovery of a stolen vehicle.

Another set of (related) diagnostic messages called Unified Diagnostic Service (UDS) messages are embodied in ISO 14229-1 (Road Vehicles - Unified Diagnostic Services) [80]. This standard specifies the requirements for diagnostic services independent of the data link connection between vehicle and remote device.

Like the J1979 OBD-II messages, UDS works to a request-response system. Particular service IDs (SIDs) are sent to the vehicle (more specifically to ECUs that support a particular service) in order to trigger a pre-determined functionality, whether that be to start a Diagnostic Management Session, interrogate UDS-compatible ECUs or reset the ECU. Again, although the standard determines what some of the UDS messages do (such as the ones given in the examples above), manufacturers are able to define their own SIDs.

7.3 OBJECTIVES

The objectives of these sets of experiments were:

- Firstly, to determine whether it was possible to transmit into the vehicle using one of these OBD-II devices;
- Secondly, whether these messages would have an effect on the vehicle (Section 7.4.1) and
- Finally, to systematically enumerate other messages that could compromise vehicle operations (Section 7.4.4).

The proof-of-concept tool (see Chapter 5) was used to systematically test a vehicle. Because of the risk of damage to vehicles, the tool was only run against a designated test vehicle.

Table 26: OBD scanning devices (dongles) tested

7.4 EXPERIMENTAL ANALYSIS

Five OBD dongles (Table 26) were connected in turn to the test vehicle, which was a small hatchback from a major manufacturer, registered in 2013. Because of the commercially sensitive nature of these messages, we have redacted identifying information about our test vehicle. Each dongle was used to test message injection into the vehicle's internal CAN bus (Figure 15).

Note that, although there are industry standard CAN tools (such as PEAK's CAN tools) for vehicle network analysis, the examination of CAN traffic was not our primary purpose. The slower and less efficient dongles (with ELM327 chips) were used because they were Bluetooth enabled. This increased the risk of compromise by adding an extraneous wireless interface (see Chapter 3.4). This aspect justified further investigation and thus gave rise to our objectives as stated in Section 7.3.

7.4.1 *Case Study: Experimental Setup*

These aftermarket devices work as an OBD to RS-232 interpreter. The signal conversion from the vehicle CAN bus to RS-232 is performed via a CAN transceiver chip and an ELM micro-controller [45]. The

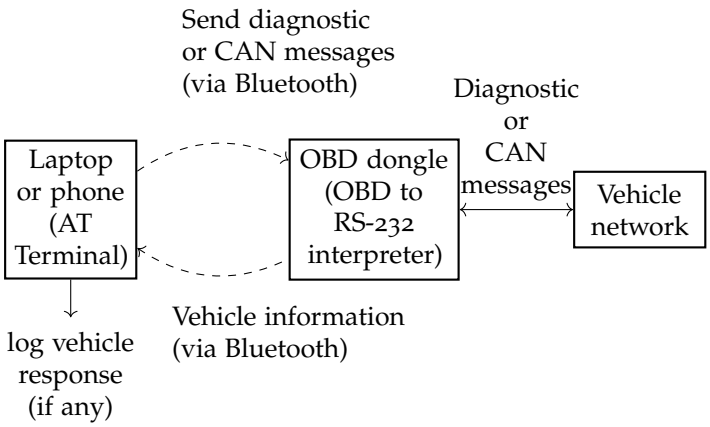

Figure 15: Vehicle with aftermarket devices attached: case study setup

ELM chip allows a serial connection to be created between data terminal equipment (DTE), such as a computer or smartphone, and data communication equipment (DCE), which are, in this case, the OBD dongles. The dongle exposes the RS-232 port via Bluetooth's Serial Port Profile (SPP) (see Chapter 3). Attention Modem (AT) commands, which are used to configure an RS-232 device, can then be sent through this serial channel via any terminal program.

7.4.2 *Case Study: Experimental Results*

Presented below are results and observations based on the experiments carried out in the case study.

The first observation of interest was that every dongle bar one (the OBDLINK MX) started broadcasting its address as soon as it was connected to the vehicle's OBD-II port. This was true regardless of whether the ignition was turned on. In the case of the OBDLINK MX, the device remained powered and could be communicated with, even with the vehicle battery at seven volts. Thus, even with the relatively small Bluetooth Class 2 range of ten metres, there is a clear security risk, especially with fixed PINs and the discoverable mode permanently enabled. Security in this regard is improved with the OBDLINK MX, where the dongle had a two minute discoverable window.

The second observation was that the low cost dongles used the more insecure legacy pairing mechanism, with the PIN being almost invariably '1234', in line with those found by [120]. The OBDLINK

MX tool asked for a comparison with a dynamically generated six-digit PIN on the phone; however, the dongle contained no screen and therefore no basis for comparison. Just selecting 'confirm' was enough to allow pairing to proceed. There was communications encryption and authentication, via the Bluetooth standard, between the serial terminal (computer or cellphone) and dongle. However, the security was greatly diminished by the fact that the PINs were short, fixed and easily determined. Furthermore, once a phone or other device was paired, it remained trusted by the dongle indefinitely and re-connected automatically once within range. The significance of this is that the phone or laptop and dongle could be pre-paired, and then planted in the vehicle afterwards.

As can be seen from Table 26, there were also several dongles that had an ELM chip version of 1.5. This is a non-existent version [45] as the version after ELM 1.4b was ELM version 2.1 and could be indicative of a counterfeit chip. Although this did not affect our experiments, the probable counterfeit ELM chips were not able to accept the full ELM `AT` command set nor did they correctly implement some of the commands.

Once devices were paired, a series of `AT` messages were sent in order to configure the dongle and monitor the traffic from whatever CAN bus was exposed on the OBD port. The appropriate messages were then sent to the dongle via a simple terminal program. The commands used are summarised in Table 27.

The modes and PIDs used in this experiment were not standard and would have been set and specified by an individual OEM. This seeming obscurity does not negate the danger as many diagnostic functions could potentially induce dangerous effects and every combination of mode and PID could be cycled through to try and brute force the combinations that could cause adverse reactions from the vehicle. Not every diagnostic message elicits a physical response or logical response, however, anything that comes back from the CAN bus could be used in future reverse engineering endeavours.

The results of trials with one non-standard diagnostic message that did produce a physical response are summarised in Table 28. The experiment was performed by sending the message both within and outside the vehicle cabin (within a five metre range) with the same results.

Table 27: Commands sent to the OBD connected dongle

Also seen from Table 28 is that sending mode and PID in a continuous stream caused functionality to flicker, rather than to stop altogether. This was probably because CAN traffic on the bus network continued to be generated on the vehicle. Essentially, messages that were sent through into the network from an external source had to contend with still flowing data. Unless the native CAN traffic is suppressed, flickering behaviour is to be expected. Furthermore, as CAN performs error checking, as soon as the messages stopped, the vehicle returned (physically) to its original state. Note that we have no knowledge of the state internally, although the diagnostic indicator of "Engine Malfunction" could signify that something within the system under test has been adversely affected.

7.4.3 *Systematic Evaluation: Experimental Setup*

Having fulfilled the first two objectives (to determine whether messages could be sent into the CAN bus and affect the vehicle), we now concentrate on what other combinations might cause a reaction. This section describes the setup of the systematic evaluation of a vehicle through an attached Bluetooth-enabled OBD-II device.

Table 28: Results of sending a non-standard diagnostic message to a test vehicle

The OBDLINK MX dongle was connected to a single test vehicle. This particular device was chosen as we could be sure that the ELM chip was not counterfeit and that it is able to accept the full `AT` command set (a full list of supported commands can be found in Sparkfun Electronics [141]). Physical setup was the same as that of the case study. From here the proof-of-concept tool used a pre-determined attack tree (shown in Figure 16) to run through the entire aftermarket device test suite, recording all outputs as described in Table 11.

Baudrate was set at 115200, which was the maximum based on the ELM device (with the default connection being set at 9600). Time intervals for all messages was set at 0.5 seconds in order to ensure that the OBD-II device had time to read and transmit the data, although this is user customisable using the proof-of-concept tool. Although a serial connection is slow compared to the speed CAN busses could operate at (40Kbit/s to 1Mbit/s for high-speed CAN for example), the set interval was enough to be able to flood the bus with enough messages to cause adverse reactions. Lower level bit-by-bit attacks, however, would not be feasible using this method.

The systematic test was first performed with ignition on (but not engine), with the assumption that, as long as the appropriate target ECUs were powered, that the vehicle would respond. Deciphering the content of the response was considered out of scope at this point in time, as we had no manufacturer CAN database available to interpret the CAN data acquired from the bus. The experiments were then repeated with the engine on with the modes and PIDs that either returned CAN data or caused a physical reaction from the vehicle.

7.4.4 *Systematic Evaluation: Experimental Results*

Recall that modes and PIDs refer to the OBD-II diagnostic test modes as embodied in the standard SAE J1979 (see Chapter 7.2.2).

Of the standard suite of modes, the vehicle returned information from the vehicle in three modes: [134]:

- `01`, which corresponds to “Show current data”;
- `06`, which corresponds to “Test results, oxygen sensor monitoring for CAN only” and
- `09`, which corresponds to “Request vehicle information”

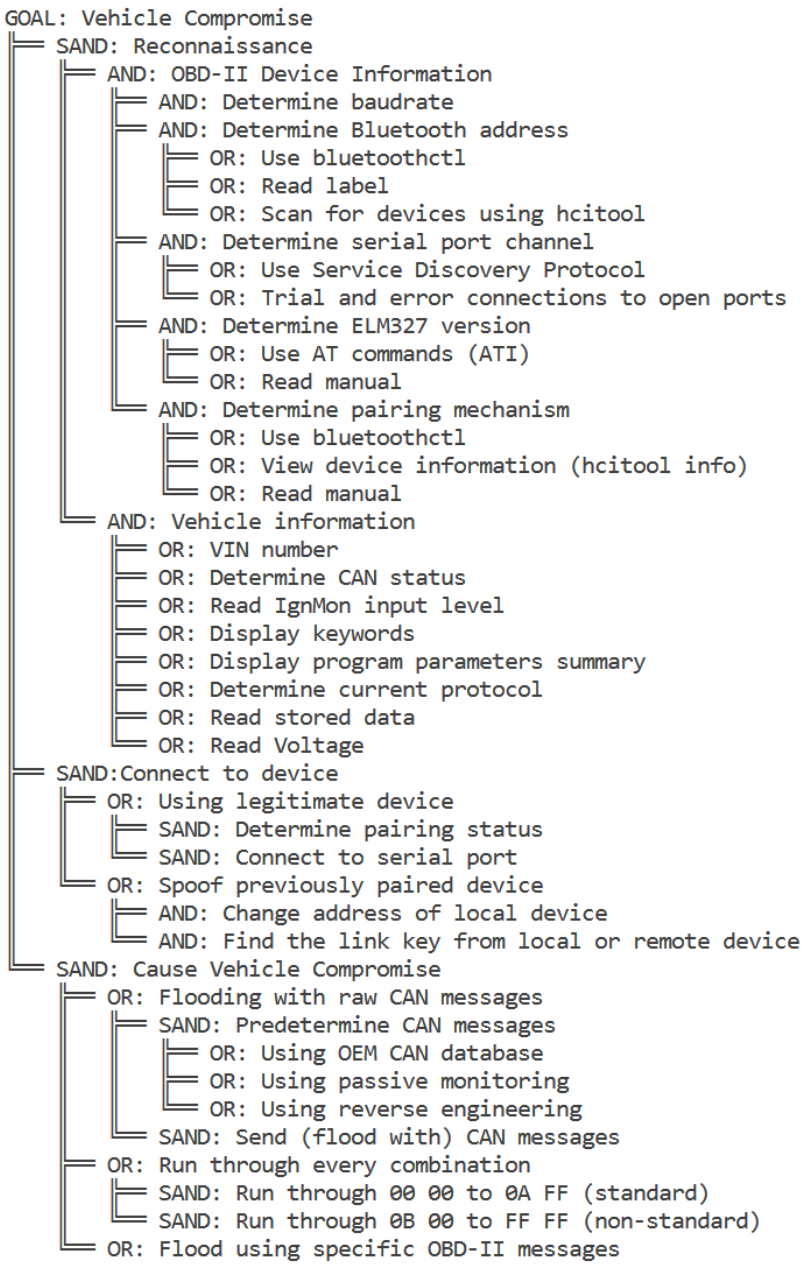

Figure 16: Attack tree used to test vehicles which have an aftermarket OBD-II device attached

Of the non-standard suite of modes, the vehicle returned information for nine different modes with all other modes returning `NO DATA`. Note that there were modes and PIDs where the vehicle returned `NO DATA`, but that there was a physical effect on the vehicle as observed in the case study (Section 7.4.2).

A summary of results can be seen in Table 29. Because non-standard modes are manufacturer dependent, the exact modes and PIDs found to affect the vehicle are not given.

Select raw CAN messages were also sent into the vehicle. These packets were pre-determined through trial and error, but consisted of messages that were known to cause an effect when trialled through a wired connection to the OBD-II port. Unlike the diagnostic messages, some of the raw CAN messages sent through only needed to be sent once to cause an effect. A summary of these results can be seen in Table 30.

7.5 ASSIGNMENT OF SEVERITY CLASSIFICATIONS

Following the same process as testing through the native headunit connection (Chapter 6.6), the severity classification was created based on findings and observations (recall that S_p is the privacy severity rating, and S_o the operational severity rating).

The table given in Figure 17 gives an overview of the severity classification for each of the results in conjunction with manual observation for tests performed with aftermarket devices.

No.	Attribute	Rating (S0-S4)	Evidence
ACT	CAN data present	Sp1,So0	Anonymous CAN data returned
POT		Sp4,So3	Profiling on CAN possible, reverse engineering, exploits could cause damage
ACT	VIN no. found	Sp2,So0	VIN number allows vehicle identification
POT		Sp4,So0	Multiple vehicle tracking possible
ACT	Caused significant reaction	Sp1,So3	Anonymous data, significant impairment
POT		Sp1,So4	Simultaneous connections to devices attached in many cars could cause multiple vehicle impairment

Figure 17: Severity classification from tests with aftermarket devices

As can be seen from the results, no personal data was acquired. This was to be expected since we are interfacing with the CAN bus and its connected ECUs, rather than the infotainment system where any personal data is most likely to be stored. Only system information was acquired, which is the reason for the S_p1 ratings. This may not hold true in a worst case scenario as CAN traffic can be used to

Table 29: Results of systematic testing. The modes and PIDs refer to the OBD-II diagnostic test modes as described in SAE J1979

Modes found	*Result*	*Description*
First mode	CAN data returned for 2 PIDs	The first PID caused the headunit screen to display "Diagnostics Mode On", the second PID caused the engine to refuse to start and the hazard lights (on the cluster only) to flash, but required message flooding. If engine is on, this causes the engine to stall. Vehicle remains unresponsive thereafter as long as message flooding continues. Once flooding stops, the instrument cluster restarts but not the engine.
Second mode	CAN data returned for 1 PID	The first PID had no physical effect, the second returned `NO DATA`, but the electronics cut out, with the instrument panel and ignition button non-responsive.
Third mode	Not recognised	`7F` returned by vehicle
Fourth mode	CAN data returned for many PIDs	12 of the PIDs each had 16 frames worth of CAN data returned by the vehicle
Fifth mode	Not recognised	`7F` returned by vehicle
Sixth to ninth mode	Not recognised	`7F` returned by vehicle

Table 30: Results of systematic testing (CAN messages)

Action	*Observation*
Unlock doors	Vehicle did not return any data, doors did not unlock, both with key in-cabin or external to the vehicle
Sending Unified Diagnostic Service messages	Hazard lights came on, "crash" was displayed on the secondary screen above the steering wheel. Vehicle doors continuously locked and unlocked. The former happened if message is sent once, the latter if message flooding is performed
Changing speed and RPM indicators on the instrument panel	CAN message flooding to this particular message ID caused the needles to fluctuate
Disabling power steering	Successful, but only if the vehicle was stationary. This message only needed to be sent once. Further message flooding had no effect.

profile individual drivers through usage patterns [49]. Therefore the potential privacy rating was set at S_p4.

The VIN number was acquired, which allows for the identification of the individual vehicle and its characteristics including make, model, year of registration, airbag type and more. With this, vehicle tracking may also be possible, as there are many online tools such as determining tax status that makes use of this number.

Finally, the last characteristic deals with whether there was significant operational effect on the vehicle. As could be seen from the case study as well as from the systematic evaluation, almost anything is possible on the vehicle should the correct CAN message be determined. Reverse engineering the right CAN message (including any diagnostic message) allows for significant operational impairment of the vehicle, hence the potential indicator of S_o3.

Potentially, many of these devices (being small and the OBD-II port hidden from general view) could be planted, and a signal sent to every device within range. This led to the assignment of the potential indicator of S_p1 (for anonymous data acquired) and S_o4 (for possible significant impairment on multiple vehicles - even simultaneously should they all be in range at once).

Like the classification in the preceding chapter, prioritisation could take place depending on the results. The edge cases of S_p0,S_o0 and any S4 rating could be dealt with straightaway. The former could be used as evidence in a security assurance case that there is low risk associated with that aspect of the component, or as evidence that there are sufficient countermeasures in place. The latter (as it affects multiple vehicles severely) would be a priority in any case.

The middle cases (S2 and S3 ratings and combinations thereof) would usually have other factors weighting it. Since we are testing the internal CAN network here, operational factors might be given precedence, as typically there is very little personal data available on the CAN bus.

Recall that validation of a black box with unknown inputs and no set of expected outputs is very difficult. The black box nature also means that formal verification is extremely challenging due to lack of knowledge of internal behaviours. Therefore validation of the classification scheme was also performed through domain expert review (reviewer biographies are available in Appendix A.2).

7.6 DISCUSSION

Like the classifications in the preceding chapter, the severity ratings can be used to assign precedence for development of counters, as evidence in security assurance cases and the rationale behind the worst-case scenario could again be used as a guideline for future tests.

The attacks that could be performed through these devices are significantly more severe then the ones identified against the headunit. This is to be expected since an attachment to the OBD-II port gives us remote access straight into the intra-vehicular CAN network. Although safety was not a factor in this instance, the effects of some of the diagnostic attacks (such as stalling the engine) are clear failure modes, and so these tests could also be used to inform safety analysis. The systematic evaluation is also able to enumerate as many manufacturer specific modes as possible, and allows a tester to observe which were most devastating to the operation (and therefore safety) of a vehicle.

The above highlights the need to consider security by design, and investigate where in the design and implementation process this could

be avoided. For example, manufacturers of devices should consider using at least Bluetooth version 2.1 to enable use of SSP.

Discoverability is also an issue, as many attacks begin by trying to enumerate the Bluetooth address. Our war-nibbling study found over a hundred vehicles broadcasting their address, even with a low-powered method (see Chapter 3). Being able to detect devices for over a minute means that exposure to (especially automated) attacks is increased, even if SSP is used. User interaction should be introduced (where it hasn't already), for example through asking the owner to actively enable Bluetooth activity so that broadcasting and duration of visibility is limited. Additionally, the number of concurrent connections to the vehicle should also be restricted unless explicitly overridden by the user.

Manufacturers should also consider hardening their architecture, so that even if an attacker manages to compromise the aftermarket device, there is no pathway to mission-critical systems. An example would be to scrutinise which CAN busses are exposed on the OBD-II port, and consider either removing the exposure, or installing an appropriately secure central gateway [159]. Application level information from a peripheral device could be secured using a secure session layer such as that proposed by [40]. The in-vehicle network could also be tested (with aftermarket devices attached) via an analysis platform or testbed such as the ones proposed by [53] to further enumerate threats.

7.7 CONCLUSION

In conclusion, a systematic evaluation enumerated manufacturer specific implementation details regarding the diagnostics port, and by injecting both OBD-II specific and raw CAN messages were able to affect the vehicle. A severity classification for these results was created and validated using domain expert review. Finally the wider issues of Bluetooth discoverability, and what manufacturers could consider was explored.

Part III

WHAT NOW?

"The future is uncertain but the end is always near."

Jim Morisson

8

TOWARDS INTEGRATING FORMAL METHODS

Empirical work in previous chapters allowed for enumeration of some of the weaknesses in vehicles, as well as the acquisition of evidence for a security assurance case. However, the real challenge is robustness and rigour. This confidence could be obtained through the use of formal methods.

In this chapter, two aspects of formal work that could be performed based on the knowledge and tools that we have built up over the previous chapters are explored. The notation and semantics that we use in this chapter are given in Section 8.1 and 8.2 respectively. Following this, we explore how information from an informal attack tree can be used to inform future design specifications via a formal analysis process (Section 8.3). Finally, we explore the concept behind the translation of an informal attack tree to a formal structure, which would allow for model based security testing (Section 8.4). This allows for further automation of the systematic security evaluation process as described in previous chapters, even if the system specifications are unknown.

8.1 NOTATION

The process algebra Communicating Sequential Processes (CSP) is used to describe the models that we use below. We choose CSP because as a process algebra it is able to represent and combine the message passing choreography expected by individual components. We give here a brief overview of the subset of CSP that we use.

Given a set of events Σ, CSP processes are defined by the following syntax:

$$P \quad ::= \quad Stop \mid e \rightarrow P \mid P_1 \,\Box\, P_2 \mid P_1 \sqcap P_2 \mid P_1 ; P_2 \mid P_1 \underset{A}{\parallel} P_2 \mid P_1 \,|||\, P_2$$

where $e \in \Sigma$ and $A \subseteq$ events.

For convenience, the set of CSP processes defined via the above syntax is denoted by CSP.

To mark the termination of a process, a special event ✓ is used.

In the above definition, the process $Stop$ is the most basic, which does not engage in any event and represents deadlock. In addition, $Skip$ is an abbreviation for $\checkmark \rightarrow Stop$. It only exhibits ✓ and then behaves as $Stop$.

The prefix $e \rightarrow P$ specifies a process that is only willing to engage in the event e, then behaves as P.

The sequential composition $P_1 ; P_2$ initially behaves as P_1 until P_1 terminates, then continues as P_2.

The generalised parallel operator $P_1 \underset{A}{\parallel} P_2$ requires P_1 and P_2 to synchronise on events in $A \cup \{\checkmark\}$. All other events are executed independently.

The interleaving operator $P_1 \mid\mid\mid P_2$ allows both P_1 and P_2 to execute concurrently and independently, except for ✓.

CSP has two mechanisms to introduce branching into a process, internal choice and external choice. The external choice $P_1 \,\Box\, P_2$ behaves either as P_1 or as P_2 with both operands made available to the environment. The internal choice $P_1 \sqcap P_2$ the choice is made non-deterministically to offer either operand.

Finally, if α is an alphabet of events, the process $CHAOS_\alpha$ behaves in the most chaotic way possible over these events. At any stage it may offer any or none of these events.

8.2 TRACE SEMANTICS

There are different semantics models for CSP processes [131]. For the purpose of this chapter, we recall the finite trace semantics.

A *trace* is a possibly empty sequence of events from Σ and may terminate with ✓. As usual, let Σ^* denote the set of all finite sequences of events from Σ, $\langle\rangle$ the empty sequence, and $tr_1 \frown tr_2$ the concatenation of two traces tr_1 and tr_2; then the set of all traces is defined as $\Sigma^{*\checkmark} = \{tr \frown en \mid tr \in \Sigma^* \wedge en \in \{\langle\rangle, \langle\checkmark\rangle\}\}$.

The trace tr_1 is a *prefix* of a trace tr_2, written as $tr_1 \leqslant tr_2$, iff $\exists\, tr' : tr_1 \frown tr' = tr_2$. Events in $A \subseteq \Sigma \cup \{\checkmark\}$ may be abstracted away from a trace tr by a hiding operator, written as $tr \setminus A$ and defined as

$$tr \setminus A = \begin{cases} \langle\rangle & \text{if } tr = \langle\rangle \\ \langle a\rangle \frown (tr' \setminus A) & \text{if } tr = \langle a\rangle \frown tr' \wedge a \notin A \\ tr' \setminus A & \text{if } tr = \langle a\rangle \frown tr' \wedge a \in A. \end{cases}$$

For convenience, when $A = \{a\}$, we shall simply write $tr \setminus a$. In general, the trace semantics of a process P is a subset $traces(P)$ of $\Sigma^{*\checkmark}$ consisting of all traces which the process may exhibit. It is formally defined recursively as follows:

$traces(Stop) = \{\langle\rangle\}$;

$traces(e \to P) = \{\langle\rangle\} \cup \{\langle e\rangle \frown tr \mid tr \in traces(P)\}$;

$traces(P_1 \,\Box\, P_2) = traces(P_1) \cup traces(P_2)$;

$traces(P_1;\, P_2) = traces(P_1) \cap \Sigma^*$
$\quad \cup \{tr_1 \frown tr_2 \mid tr_1 \frown \langle\checkmark\rangle \in traces(P_1) \wedge tr_2 \in traces(P_2)\}$;

$traces(P_1 \underset{A}{\parallel} P_2) = \{tr \in tr_1 \underset{A}{\parallel} tr_2 \mid tr_1 \in traces(P_1) \wedge tr_2 \in traces(P_2)\}$
where $tr_1 \underset{A}{\parallel} tr_2 = tr_2 \underset{A}{\parallel} tr_1$ is defined as follows with $a, a' \in A$ and $b, b' \notin A$:

$\langle\rangle \underset{A}{\parallel} \langle\rangle = \{\langle\rangle\}$; $\langle\rangle \underset{A}{\parallel} \langle a\rangle = \emptyset$;$\langle\rangle \underset{A}{\parallel} \langle b\rangle = \{\langle b\rangle\}$;

$\langle a\rangle \frown tr_1 \underset{A}{\parallel} \langle b\rangle \frown tr_2 = \{\langle b\rangle \frown tr \mid tr \in \langle a\rangle \frown tr_1 \underset{A}{\parallel} tr_2\}$;

$\langle a\rangle \frown tr_1 \underset{A}{\parallel} \langle a\rangle \frown tr_2 = \{\langle a\rangle \frown tr \mid tr \in tr_1 \underset{A}{\parallel} tr_2\}$

$\langle a\rangle \frown tr_1 \underset{A}{\parallel} \langle a'\rangle \frown tr_2 = \emptyset$ where $a \neq a'$;

$$\langle b\rangle \frown tr_1 \underset{A}{\parallel} \langle b'\rangle \frown tr_2 \quad = \quad \{\langle b\rangle \frown tr \mid tr \in tr_1 \underset{A}{\parallel} \langle b'\rangle \frown tr_2\} \cup \{\langle b'\rangle \frown tr \mid tr \in \langle b\rangle \frown tr_1 \underset{A}{\parallel} tr_2\}$$

$traces(P_1 \,|||\, P_2) = \{tr \in tr_1 \,||| \, tr_2 \mid tr_1 \in traces(P_1) \wedge tr_2 \in traces(P_2)\}$
where $tr_1 \,||| \, tr_2 = tr_1 \underset{\emptyset}{\parallel} tr_2$.

Therefore, $traces(P_1 \,|||\, P_2) = traces(P_1 \underset{\emptyset}{\parallel} P_2)$.

A process P is said to *trace-refine* a process Q (written $Q \sqsubseteq_T P$) if $traces(P) \subseteq traces(Q)$. There are other flavors of refinement, but we restrict ourselves to trace refinement below.

8.3 INTEGRATION INTO FUTURE DESIGN

The context of the work in this section is the way in which various components are combined to achieve the final vehicle product.

Components are often generic, with many general purpose features. This promotes their reuse, which drives overall costs within the supply chain down. Larger components are often provided as whole "off-the-shelf" (OTS) subsystems (for example, an infotainment unit), with each component therein originating with a different manufacturer.

Within the automotive supply chain, system integrators often do not have the final detailed designs of the components, especially where these components represent intellectual property such as source code.

One proposed solution to the problem that integrators face when trying to build secure systems out of unknown components is based on testing [47]. In this case, though, testing cannot be a full solution as the component output in response to an input may not be well defined. For example, acknowledging, ignoring or rejecting the input may all present in the same way. Part of this problem could be due to the testing interface, however, absent or loose specifications could also lead to ambiguity in terms of what the response should actually be. In some cases, components also include behaviours that cause the larger system to be insecure. Furthermore, as each component of each tier of supplier integrates with another subsystem, another layer of obscurity is added with regards to the overall system.

Discussed in this section is a methodology that uses the systematic and semi-automated penetration testing process as described in preceding chapters in order to identify additional security requirements. These requirements are over and above the functional and integrational requirements that already exist for the system, and can be used to improve the design of the system with respect to security.

The motive for beginning the process with testing is to acquire confidence with regard to the overall implementation. The testing process moves knowledge of the component along the black-white spectrum, where we can then extract requirements for secure behaviour in the given context to help negate security flaws in subsequent design processes. This is particularly valuable where a system contains many third party components of which even the manufacturer may not have complete sight, because of commercial sensitivities.

8.3.1 *Method Overview*

In this section we present our proposed methodology for combining third party components securely. An overview of the methodology is given in Figure 18. Since the example in Section 8.3.2 is that of an infotainment unit being accessed over its Bluetooth interface, the first and last boxes in figure 18, which represent the starting and ending states of the system, are populated with a simplified infotainment

unit (comprising an operating system and its Bluetooth interface) and its updated version respectively.

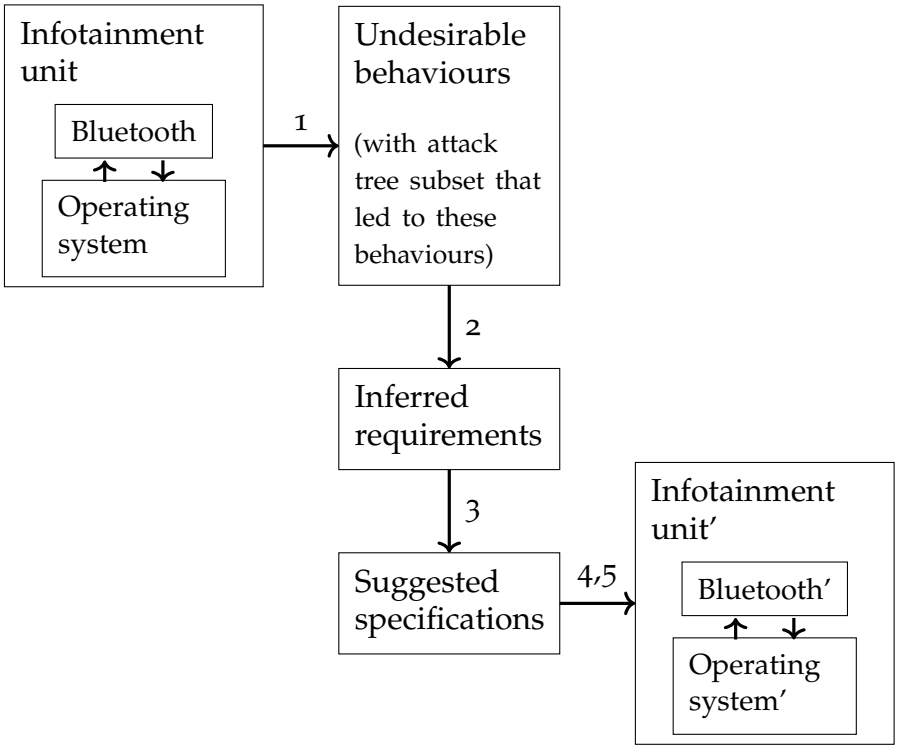

Figure 18: Overall proposed methodology for integration into future design. The numbers refer to their respective steps in list below

The methodology has four steps, that correspond to the numbers 1-4 in Figure 18. The steps are further detailed below.

8.3.1.1 *Step 1: Security Testing*

Recall that full functional specifications are unlikely to be available in the automotive domain due to concerns regarding intellectual property. Additionally, because of the tiered supply chain common in the automotive industry, systems become even more obscured with every tier up the supply chain. It is for this reason that we begin with security testing. In the context of this chapter, initial attack trees are first defined, with the root of every attack tree being an attack goal. These goals can be as low level (flood an open port with appropriate data) or high level (denial of service) as needed and tailored to the target interface or component.

8.3.1.2 *Step 2: Inferring requirements*

Requirements can be "inferred" from whichever attack proved successful. This is a process of inference, and is essentially a negation of observed undesirable behaviours from the testing process above. The determination of security requirements at this stage can be traced and cross-referenced back to the attack tree. This allows for specific insecurities to be addressed (even if we did not possess full functional specifications) as well as separation of security requirements from other types of requirements, which is known to be useful for interaction analysis [95], [130].

8.3.1.3 *Step 3: Suggesting specifications*

Once the requirements gathering phase is considered complete, possible specifications could be suggested using an automated process such as design space exploration [58]. There may be a number of different design choices (and therefore specifications) that could be made to mitigate the threat. These derived specifications could be cross-referenced with other subsets of specifications (such as function or safety), and where there are contradictions, could help clarify design choices depending on what could be deemed an acceptable risk.

Where there are no conflicts, whatever derived security specifications from our process could be added to the overall set of specifications. If this is the case, each of the suggested security-focused design choices could also be cross-referenced to the attack tree to tackle the root of the problem (see case study in Section 8.3.2).

8.3.1.4 *Step 4: Incorporating into design*

Once specifications have been agreed upon, there are several options available. These specifications could be sent onto a tier one manufacturer (or a supplier further down the supply chain) for them to address or incorporate. Alternatively, the end user could follow up with model-based design and testing processes if they are already being carried out in-house. We discuss the latter within the context of our case study. The reason for keeping this step flexible is to enable incorporation of this methodology into the wider processes that might be carried out by the designer, OEM or analyst, which may help save on time and cost.

8.3.1.5 *Step 5: Verification*

Since we now have a specification of the security measures we want to have in place (or a negation of undesirable behaviours), we can use formal methods to verify against a model. For example, we could verify that our suggested inferred security specification prevents the system from entering an undesired state. If verified, this means that that particular undesired behaviour has been overcome (see Section 8.3.2.5).

8.3.2 *Case Study*

Within the automotive industry, OEMs integrate various bought-in systems in order to create the working vehicle. For this paper we concentrate on the infotainment unit, where various diverse technologies (including external facing interfaces such as Bluetooth) are integrated to deliver functionality such as hands-free communication, radio and satellite navigation. A generic head unit would typically comprise a System-on-a-Chip (SoC), containing the operating system, some memory, chips for wireless communications (such as a Bluetooth chip), connections for other interfaces such as Universal Serial Bus (USB), antennas and transceivers for Controller Area Network (CAN) bus data (see Figure 19).

We demonstrate the proposed methodology discussed above using a case study below. Although this case study came from a single vehicle, it can be reasonably assumed that vehicles of the same make, model and age would share the same weaknesses and vulnerabilities as production lines are standardised.

8.3.2.1 *Step 1: Security Testing*

We tested eight vehicles empirically (see Chapter 6), from which a number of undesirable behaviours were found, including, in two cases, the ability to mount the filesystem of the infotainment unit with read and write access. This was possible because of the presence of the OBEXFTP service (see Chapter 3). This behaviour is highlighted as an example for this case study.

There are three possible usages covered under the File Transfer Profile (FTP) [21]:

- Browsing of the object store (i.e. the filesystem) of another Bluetooth device

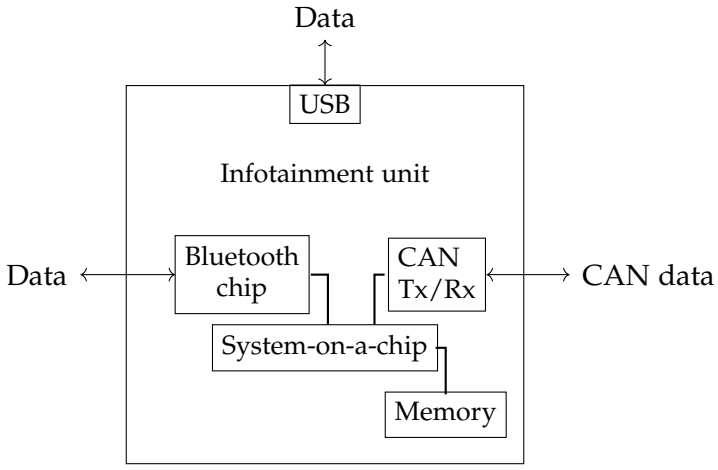

Figure 19: The contents of a generic automotive infotainment unit

- Transfer of objects between Bluetooth devices
- Manipulation of objects on another Bluetooth device (including deletion and creation of objects)

The workings of the profile are based on a client-server process, with the client being the device that initiates the process, pulls or pushes objects to and from a server, or instructs the server on actions to perform on objects. The server is responsible for providing the object exchange (OBEX) server and folder browsing capabilities.

It is worth noting that the latest spec for FTP is from 2015, but that the software implemented on a vehicle is often older than the registration year of the vehicle, the implication to this being that it may not be the latest (or most secure) software that is currently on the vehicle.

Another important distinction is that we are not attacking the Bluetooth protocol (or any other protocols involved between the Bluetooth stack and the operating system) in any way. There was no compromise of the pairing mechanism, the operating system or of the initiating or remote Bluetooth enabled devices. The behaviour was completely legitimate and in keeping with the functionality of the Bluetooth protocol.

Additionally, all users of the Bluetooth system in this test vehicle (and more generally across all vehicles) have the same authorisation levels. In other words, all services are available to all paired and connected users regardless of who they are (although support for differing levels of access exists). Authentication thus becomes irrele-

vant, with authorisation simply comprising services offered to either paired or unpaired devices.

8.3.2.2 *Step 2: Inferring Requirements*

Consider the case study and the steps that led to being able to mount the infotainment unit's filesystem.

First, reconnaissance is performed to determine the characteristics of the Bluetooth interface, including its Bluetooth address and the services it offers. A summary of these characteristics from this particular test vehicle is given in (Chapter 6, Table 16).

A connection was then made to the interface using a legitimate pairing and device (that is to say that the connection, vehicle or device had not been tampered with in any way), and mounted the file system.

Since being able to mount the filesystem and read or manipulate objects is undesirable, our inferred requirement from this would be "no unauthorised external agency should be able to see or influence the operating system's filesystem".

8.3.2.3 *Step 3: Suggesting Specifications*

Being able to mount the filesystem through Bluetooth could lead to injection of malware, directory traversal and data extraction, manipulation or destruction. Each of these dangers (leaf nodes) could be prevented by not allowing for the mounting (at least on the Bluetooth interface) in the first place. Based on our attack tree, the design decision could be narrowed to removing the implementation of the OBEXFTP service so that this action is not possible (and thereby ruling out all leaf nodes). If there is a genuine need for such functionality, then design decisions could be expanded to mitigate each of the lower tier of actions that could lead to compromise.

Thus, based on the case study attack tree (Figure 20), we could either create specifications that:

- remove the ability to request files or data (which might conflict with functional requirements of the infotainment unit)
- remove the ability to mount the filesystem (by removing the OBEXFTP service, which, likewise, might have functional or cost implications)

- or allow the above, but remove support for extracting, deleting or creating (injecting) files (which would conflict with the required functionality of the FTP server role as specified by Bluetooth SIG [21]).

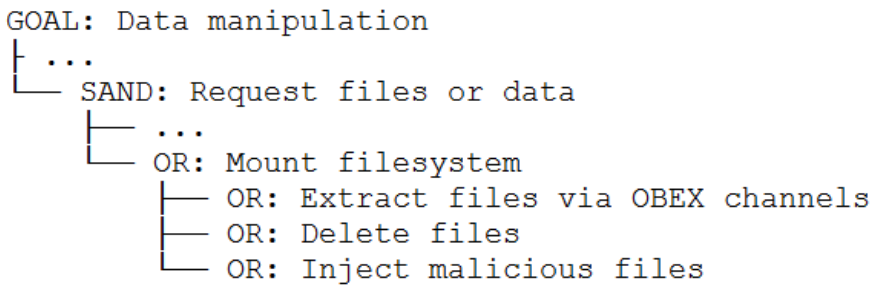

Figure 20: Attack tree: a representative example for this case study [35]

Here a small example is presented, however, the number of suggestions when scaling up could be beyond manual means. In these cases a mechanical method of choice, such as design space exploration [58], could be used instead. This would require further research that is currently beyond the scope of this study.

8.3.2.4 *Step 4: Model-based Design*

There have already been studies which formally analyse the Bluetooth protocol with regards to authentication and secrecy properties [6], [32], [123]. However, it is not the protocol being attacked, but rather it is a probe into the larger system in which it resides, using a legitimate connection and a legitimate device. The Bluetooth specification would not fail on its own. This is an example of two components in themselves being secure, but exhibiting insecure behaviour when combined into a larger system. The kind of analysis that would therefore be more appropriate is a reachability analysis, to demonstrate that such an insecure system state could not be reached through specific pathways (specifically the pathways dictated by the attack tree).

The specification for the Bluetooth FTP is available [21] and so we develop a small illustrative CSP model:

channel $pair, connect, advertise\_service, service\_discovery$

channel $service1, service2, displayfs$

channel $obexftp : OBEXFTP\_CMD$

datatype $OBEXFTP\_CMD =$

$Selectserver \mid NavigateFolder \mid MountFS \mid Push \mid Pull \mid$

$$
\begin{aligned}
&\quad CreateFolder \mid Copy \mid Move \mid Rename \mid Delete \mid SetPermission \\
&BT = pair \rightarrow connect \rightarrow advertise\_service \rightarrow OFFER\_SERVICE \\
&OFFER\_SERVICE = \\
&\qquad service1 \rightarrow OFFER\_SERVICE \\
&\quad \Box\ service2 \rightarrow OFFER\_SERVICE \\
&\quad \Box\ .. \\
&\quad \Box\ obexftp?cmd \\
&\qquad \text{if } cmd == MountFS \text{ then} \\
&\qquad\quad displayfs \rightarrow OFFER\_SERVICE \\
&\qquad \text{else } OFFER\_SERVICE \\
&USER = pair \rightarrow connect \rightarrow service\_discovery \rightarrow USE\_SERVICE \\
&USE\_SERVICE = \\
&\qquad service1 \rightarrow USE\_SERVICE \\
&\quad \sqcap\ service2 \rightarrow USE\_SERVICE \\
&\quad \sqcap\ .. \\
&\quad \sqcap\ obexftp!mountfs \rightarrow displayfs \rightarrow USE\_SERVICE \\
&IMPL = BT \underset{\alpha_{BT} \cap \alpha_{USER}}{\parallel} USER
\end{aligned}
$$

For details on the mechanisms behind pairing as well as information on service profiles, please refer to Chapter 3.

8.3.2.5 *Formal Verification*

We then developed a suggested specification from our inferred requirement, such as never displaying the filesystem:

$SPEC = CHAOS_{\alpha_{IMPL} \setminus \{displayfs\}}$

assert $SPEC \sqsubseteq IMPL$

Result: FAIL (using FDR [150], see Figure 21)

which fails during the analysis process (Figure 21). If, however, we removed the OBEXFTP service (which was one of the design options discussed in Section 8.3.2.3), and assuming none of the other services offered the ability to mount a filesystem, it would verify correctly.

This exercise is used to show that the inferred requirement is not met by the standard Bluetooth FTP specification. In fact, the Bluetooth FTP specification (that a server must respond to a request from a client to respond to requests for "Folder Listing Objects") is contradictory to our inferred requirement. Because of this contradiction, any attempt to remove support whilst still maintaining the profile would be breaking Bluetooth's specification. Additional countermeasures introduced outside of the Bluetooth specification to mitigate this insecurity may introduce new dangers.

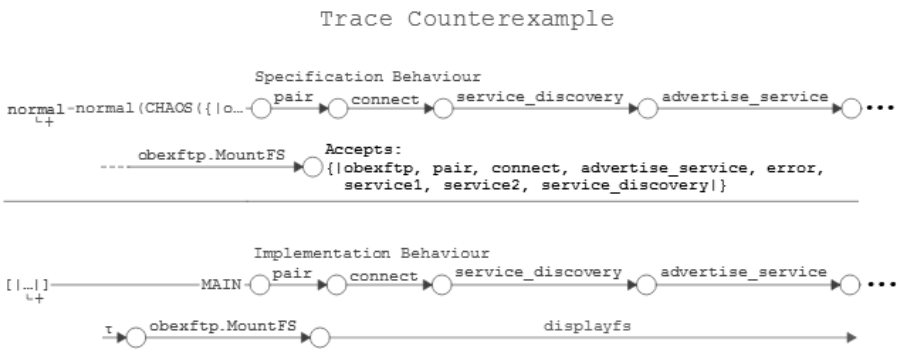

Figure 21: FDR trace of the assertion on our Bluetooth FTP model [35], ending in failure

In this case, the model and example specification is simplistic enough to make it self-evident that the removal of OBEXFTP would allow for successful verification. Circumstances under which this exercise would be of additional value include:

- where there are sufficiently complex systems;
- where there is more data or information regarding the inner workings of the system under investigation to create a more accurate model, or
- where there is more than one path to mount the filesystem (or, more generally, to achieve any other undesirable behaviour).

8.3.3 *Discussion*

This methodology has many benefits. It is suited for tiered supply chains, as there is no need to have the complete specification of the integrated item in order to test for security concerns and, furthermore, reflects real world security issues that have arisen through the testing process. The attack tree methodology allows for systematism, traceability and documentation, especially where design choices are concerned. These design choices could also be cross-referenced against scenarios that were posited in attack trees but were not tested. Any security requirements gathered can be kept separate for interaction analysis and allows for reasoning about alternatives. The formal exercise could allow for clarification of ambiguities, and using a verifier leads to a higher level of confidence in the resulting design (albeit dependent on the trustworthiness of the model constructed).

Limitations include the fact that the initial creation of the attack tree is manually guided due to the black-box nature of testing (as discussed in Chapter 4). This can be mitigated through domain expert input in reviewing the tree as well as through repeated testing over various makes and models of vehicles. There is also a one-off cost of building these attack trees, although these trees could be re-used in any future design and testing process.

As testing scope or domain knowledge expands, the trees themselves could also become numerous and crowded, and to that end, work could be done to index or navigate these trees (already searchable through known algorithms such as breadth first and depth first). Problems with scalability could also be mitigated using mechanical tools such as those associated with design space exploration. A further limitation is the data available to construct the model at the end of the process which directly impacts the quality of the model created. However, we envisage that analysts, designers and other end users would either send the specific security requirements to the appropriate supplier, or would themselves have the information required to construct a more thorough model.

8.3.4 *Summary*

In this section, a methodology for securely combining third party components into a wider system is presented, using the results of systematic security evaluation as discussed above. Furthermore, a demonstration of the application of this methodology is presented, by applying it in the example context of an automotive head unit using Bluetooth as the test interface. Weaknesses were found through a structured security testing process. Using the case study of being able to mount the filesystem through Bluetooth, there is also a demonstration on how to infer security requirements and suggest specifications. Both can be cross-referenced to other requirements, and traced back using attack trees. A possible follow-on process is also illustrated. We envision that end users such as security analysts would find this constructive in fulfilling their goals to improve and strengthen the security of their systems.

8.4 TRANSLATION OF INFORMAL ATTACK TREES

The attack trees used in previous chapters were informal and manually created. However, formality is possible, with the foundations of formal descriptions laid by [108] and their process algebraic nature recognised in works such as [154]. Recall that automotive specific attack trees have been discussed in literature; for example, [136] looked at attack tree generation and gave formal descriptions of the trees. This is orthogonal to our research (in translating low level attack trees). In our case, the trees have already been pre-built based on reconnaissance of a black box system, rather than the automatic generation of an attack tree from a fully specified system under test. Recall also that model-based security testing (MBST) focuses on testing security properties, security mechanisms and system environments (see Chapter 2).

8.4.1 *Background*

The conventional attack tree has conjunction (`AND`) and disjunction (`OR`) operators, but extensions have since been proposed with the sequential conjunction (`SAND`) operator [83] (see Chapter 4.1 for more information on attack trees). There are two types of ordering: time dependent or condition dependent. In this chapter, we adopt the condition dependent paradigm.

Following the formalisation of attack trees given in [83] and [108], if $\mathbb{A}$ is the set of possible atomic attacker actions, the elements of the attack tree $\mathbb{T}$ are $\mathbb{A} \cup \{OR, AND, SAND\}$, and an attack tree is generated by the following grammar, where $a \in \mathbb{A}$:

$$t ::= a \mid OR(t, \ldots, t) \mid AND(t, \ldots, t) \mid SAND(t, \ldots, t)$$

Attack tree semantics have also been defined by interpreting the attack tree as a set of series-parallel (SP) graphs [83]. Definition of SP graphs requires first the definition of source-sink graphs and here we use the definitions from [83].

Definition 1: A source-sink graph over $\mathbb{A}$ is a tuple $G = (V, E, s, z)$ where V is a set of vertices, E is a multiset of edges with support $E* \subseteq V \times \mathbb{A} \times V$, $s \in V$ is a unique source and $z \in V$ is a unique sink, and $s \neq z$.

The sequential composition of G and another graph G', denoted by $G \cdot G'$ results from the disjoint union of G and G' and linking the sink of G with the source of G'. Thus, if $\dot\cup$ denotes the disjoint union and $E^{[s/z]}$ denotes the multiset of E where vertices z are replaced by s, then $G.G'$ can be defined as:

$$G \cdot G' = (V \setminus \{z\} \dot\cup V', E^{[s'/z']} \dot\cup E', s, z')$$

Parallel composition, denoted by $G \parallel G'$ is similar (differing only in that two sources and two sinks are identified) and can be defined as:

$$G \parallel G' = (V \setminus \{s, z\} \dot\cup V', E^{[s'/s, z'/z]} \dot\cup E', s', z')$$

Definition 2: The set $\mathbb{G}_{\mathcal{SP}}$ over $\mathbb{A}$ is defined inductively by:

For $a \in \mathbb{A}$, $\xrightarrow{a}$ is an SP graph,

If G and G' are SP graphs, then so are $G \cdot G'$ and $G \parallel G'$.

Hence, the full SP graph semantics for attack tree $\mathbb{T}$ can be given by the function:

$$[\![\cdot]\!]_{\mathcal{SP}} : \mathbb{T} \to \mathcal{P}(\mathbb{G}_{\mathcal{SP}})$$

This is defined recursively. If $a \in \mathbb{A}$, $t_i \in \mathbb{T}$, and $1 \leqslant i \leqslant k$, then

$[\![a]\!]_{\mathcal{SP}} = \{\xrightarrow{a}\}$
$[\![OR(t_1, \ldots, t_k)]\!]_{\mathcal{SP}} = \bigcup_{i=1}^{k} [\![t_i]\!]_{\mathcal{SP}}$
$[\![AND(t_1, \ldots, t_k)]\!]_{\mathcal{SP}} = \{G_1 \parallel \ldots \parallel G_k \mid (G_1, \ldots, G_k) \in [\![t_1]\!]_{\mathcal{SP}} \times \ldots \times [\![t_k]\!]_{\mathcal{SP}}\}$
$[\![SAND(t_1, \ldots, t_k)]\!]_{\mathcal{SP}} = \{G_1 \cdot \ldots \cdot G_k \mid (G_1, \ldots, G_k) \in [\![t_1]\!]_{\mathcal{SP}} \times \ldots \times [\![t_k]\!]_{\mathcal{SP}}\}$
where $[\![t]\!]_{\mathcal{SP}} = \{G_1, \ldots, G_k\}$ corresponds to a set of possible attacks G_i

Since, in this thesis, the construction of the attack tree is based on penetration testing techniques (see Chapter 4), all leaves on the tree can be considered actions. The combination of these actions can be translated into the processes that form part of a test case. This is conducive to the use of process algebra such as Communicating Sequential Processes (CSP) and furthermore the equivalence of the semantics (see Section 8.4.3) means that we can use synonymous operators to transform a pre-built attack tree.

8.4.2 *Methodology*

In this section, an overview of the methodology (Figure 22) is presented. Recall that vehicle manufacturers often incorporate off-the-shelf (OTS) components into their work, and their specifications are not always available. Manufacturers thus have to approach testing with this uncertainty.

We begin by assuming a System under Test (SUT), which may be either an OTS component or contain one. We also assume the existence of a corresponding attack tree (see Figure 22). Section 8.4.5 illustrates our approach with an attack tree developed for a vehicle network that includes a Bluetooth connection. It is worth observing here that an attack tree for Bluetooth systems essentially systematises the known attacks on Bluetooth, and therefore although it may be updated as new attacks are constructed, the development of the attack tree is a one-off cost. The same attack tree will work for any Automotive Bluetooth system.

If a specification (or abstract model) of the SUT is available we use it, but in many cases (including the example in Section 8.4.5) the specification of the SUT is confidential or contains confidential components. In this case we can under-approximate a specification to begin the process. Successive iterations of the process allow us to refine this under-approximation. We illustrate this under-approximation in Section 8.4.4. The approach is to generate tests against this model. Tests are automatically generated using FDR (the refinement checker for CSP) and the specification and the attack tree are then compared. Each possible route through the attack tree represents a potential attack, and the Test Case Generator compiles a list of all the attacks that the specification permits. Note that in the case of an under-approximated specification all possible attacks will be permitted.

The next step is to convert the formal tests into implementation tests. This process is detailed in Section 8.4.4.2. Note that these implementation tests are in fact attacks (or potential attacks) on the SUT. Not all the test implementation tests generated from an attack tree can be fully automated. The attack tree may contain nodes that require manual input, in which case the implementation tests will require (partial) manual interaction. The ones that do not require manual input may be executed directly on a testbed. The report contains

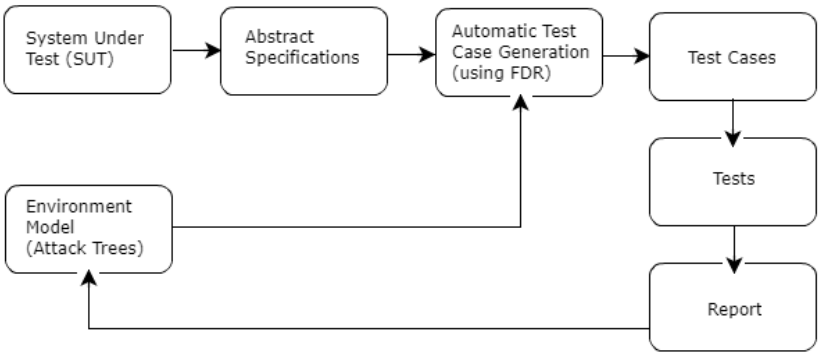

Figure 22: Using FDR and attack trees to generate test cases

the test results. Since tests are really attacks on the SUT, we consider a test to be successful if the attack succeeds.

In the remainder of this section we present in more detail the transformation of attack trees to CSP processes (Section 8.4.3), as well as a proof of the equivalence of the semantic models.

8.4.3 *Transforming Attack Trees into CSP Processes*

In this section we use CSP to describe the attack tree. We choose CSP because as a process algebra it is able to represent and combine the actions of the attack tree into a set of processes that could subsequently be used for test case generation.

In principle, the logic gates of the attack tree can be considered CSP operators [69] as follows:

- Since the `AND` logic gate demands that all actions must be successful for the branch to be considered complete, the interleave operator (|||) is used. This operator joins processes that operate concurrently but without them necessarily interacting or synchronising.
- The sequential composition operator (;) is used for the `SAND` logic gate. The former echoes the `SAND` logic gate, in that the first process must terminate successfully before the next is allowed;
- The external choice operator (□) (where any process could be chosen dependent on the environment in which it operates) is used for the `OR` logic gate.

Thus formally, we define the following transformation function trans : $\mathbb{T}_{SAND} \to CSP$ where $\Sigma = \mathbb{A}$:

$\text{trans}(a) = a \to \mathit{Skip}$ for $a \in \mathbb{A}$;

$\text{trans}(OR(t_1, \ldots, t_n)) = \text{trans}(t_1) \,\Box\, \ldots \,\Box\, \text{trans}(t_n)$;

$\text{trans}(AND(t_1, \ldots, t_n)) = \text{trans}(t_1) \,|||\, \ldots \,|||\, \text{trans}(t_n)$;

$\text{trans}(SAND(t_1, \ldots, t_n)) = \text{trans}(t_1)\,;\, \ldots ;\, \text{trans}(t_n)$;

In order to show the correctness of the above transformation, it is necessary to make the two semantics in $\mathbb{G}_{\mathcal{SP}}$ and $\Sigma^{*\checkmark}$ compatible for comparison. Recall from [83] that each SP graph represents a possible way to carry out an attack. In such a graph, an AND vertex indicates that actions along its branches must be executed. However, there is no restriction on the order of their executions. In other words, their executions are interleaving in general. Therefore, it is possible to serialise the actions from a SP graph, where parallel compositions of graphs is considered as interleaving and sequential composition as concatenation.

Given G in $\mathbb{G}_{\mathcal{SP}}$, let $\text{serials}(G)$ denote the set of all possible ways to serialise G, which is formally defined as follows:

$\text{serials}(\xrightarrow{a}) = \{\langle a \rangle\}$;

$\text{serials}(G_1 \parallel G_2) = \{tr \in tr_1 \,|||\, tr_2 \mid tr_1 \in \text{serials}(G_1) \wedge tr_2 \in \text{serials}(G_2)\}$;

$\text{serials}(G_1 \cdot G_2) = \{tr_1 \frown tr_2 \mid tr_1 \in \text{serials}(G_1) \wedge tr_2 \in \text{serials}(G_2)\}$.

For convenience, we denote the set of all prefixes from $\text{serials}(G)$ by $\text{pserials}(G)$, i.e., $\text{pserials}(G) = \{tr \mid \exists\, tr' \in \text{serials}(G) : tr \leqslant tr'\}$. We also denote $\text{pserials}(t) = \bigcup_{G \in [\![t]\!]_{\mathcal{SP}}} \text{pserials}(G)$ for all attack trees $t \in \mathbb{T}_{SAND}$.

As SP graphs have no ✓ action, we shall hide ✓ when comparing the semantics of SP graphs with CSP processes. Therefore, we denote $traces(P) \setminus \checkmark = \{tr \setminus \{\checkmark\} \mid tr \in traces(P)\}$.

The correctness of transforming attack trees into CSP processes is guaranteed by the following result: $\forall\, t \in \mathbb{T}_{SAND}, \text{pserials}(t) = traces(\text{trans}(t)) \setminus \checkmark$.

The proof is done by induction on the structure of t.

Base case: Consider $t = a$; then, $[\![t]\!]_{\mathcal{SP}} = \{\xrightarrow{a}\}$ and $\text{pserials}(t) = \{\langle\rangle, \langle a \rangle\}$. We also have $\text{trans}(t) = a \to \mathit{Skip}$ and $traces(\text{trans}(t)) = \{\langle\rangle, \langle a \rangle, \langle a, \checkmark \rangle\}$. Hence, it is straightforward that $\text{pserials}(t) = traces(\text{trans}(t)) \setminus \checkmark$.

Induction step:

CASE $t = OR(t_1, \ldots, t_n)$: It is straightforward that

- $[\![t]\!]_{\mathcal{SP}} = \bigcup_{i=1,\ldots,n} [\![t_i]\!]_{\mathcal{SP}}$, and
- $\mathit{traces}(\mathsf{trans}(t)) = \mathit{traces}(\mathsf{trans}(t_1) \,\Box\, \ldots \,\Box\, \mathsf{trans}(t_i))$.

Then, we have

$$\begin{aligned}
tr \in \mathsf{pserials}(t) \quad &\Leftrightarrow \quad \exists\, G \in [\![t]\!]_{\mathcal{SP}} : tr \in \mathsf{pserials}(G) \\
&\Leftrightarrow \quad \exists\, i \in \{1,\ldots,n\}, G \in [\![t_i]\!]_{\mathcal{SP}} : tr \in \mathsf{pserials}(G) \\
&\Leftrightarrow \quad \exists\, i \in \{1,\ldots,n\} : tr \in \mathsf{pserials}(t_i) \\
&\Leftrightarrow \quad tr \in \mathit{traces}(\mathsf{trans}(t_i)) \setminus \checkmark \text{ by induction hypothesis} \\
&\Leftrightarrow \quad tr \in \mathit{traces}(\mathsf{trans}(t)) \setminus \checkmark.
\end{aligned}$$

CASE $t = AND(t_1, \ldots, t_n)$: It is obvious that:

- $[\![t]\!]_{\mathcal{SP}} = \{G_1 \parallel \ldots \parallel G_n \mid G_i \in [\![t_i]\!]_{\mathcal{SP}} \forall\, i = 1,\ldots,n\}$, and
- $\mathit{traces}(\mathsf{trans}(t)) = \mathit{traces}(\mathsf{trans}(t_1) \,|||\, \ldots \,|||\, \mathsf{trans}(t_i))$.

Then, we have

$$\begin{aligned}
tr \in \mathsf{pserials}(t) \quad &\Leftrightarrow \quad \exists\, G \in [\![t]\!]_{\mathcal{SP}} : tr \in \mathsf{pserials}(G) \\
&\Leftrightarrow \quad \forall\, i \in \{1,\ldots,n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}} : \\
&\qquad\quad tr \in \mathsf{pserials}(G_1 \parallel \ldots \parallel G_n) \\
&\Leftrightarrow \quad \forall\, i \in \{1,\ldots,n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}}, tr' \in \mathsf{serials}(G_1 \parallel \ldots \parallel G_n) : \\
&\qquad\quad tr \leqslant tr' \\
&\Leftrightarrow \quad \forall\, i \in \{1,\ldots,n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}}, tr_i \in \mathsf{serials}(G_i) : \\
&\qquad\quad tr' \in tr_1 \,|||\, \ldots \,|||\, t_n \wedge tr \leqslant tr' \\
&\Leftrightarrow \quad \forall\, i \in \{1,\ldots,n\}, \exists\, tr_i \in \mathit{traces}(\mathsf{trans}(t_i)) \setminus \checkmark : \\
&\qquad\quad tr' \in tr_1 \,|||\, \ldots \,|||\, t_n \wedge tr \leqslant tr' \text{ by induction hypothesis} \\
&\Leftrightarrow \quad \forall\, i \in \{1,\ldots,n\}, \exists\, tr_i' \leqslant tr_i \in \mathit{traces}(\mathsf{trans}(t_i)) \setminus \checkmark : \\
&\qquad\quad tr \in tr_1' \,|||\, \ldots \,|||\, t_n' \\
&\Leftrightarrow \quad tr \in \mathit{traces}(\mathsf{trans}(t)) \setminus \checkmark.
\end{aligned}$$

CASE $t = SAND(t_1, \ldots, t_n)$: It is obvious that:

- $[\![t]\!]_{\mathcal{SP}} = \{G_1 \cdot \ldots \cdot G_n \mid G_i \in [\![t_i]\!]_{\mathcal{SP}} \forall\, i = 1,\ldots,n\}$, and
- $\mathit{traces}(\mathsf{trans}(t)) = \mathit{traces}(\mathsf{trans}(t_1);\, \ldots;\, \mathsf{trans}(t_n))$.

Then we have:

$$
\begin{aligned}
tr \in \text{pserials}(t) \quad &\Leftrightarrow \quad \exists\, G \in [\![t]\!]_{\mathcal{SP}} : tr \in \text{pserials}(G) \\
&\Leftrightarrow \quad \forall\, i \in \{1, \ldots, n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}} : \\
&\qquad tr \in \text{pserials}(G_1 \cdot \ldots \cdot G_n) \\
&\Leftrightarrow \quad \forall\, i \in \{1, \ldots, n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}}, tr' \in \text{serials}(G_1 \cdot \ldots \cdot G_n) : \\
&\qquad tr \leqslant tr' \\
&\Leftrightarrow \quad \forall\, i \in \{1, \ldots, n\}, \exists\, G_i \in [\![t_i]\!]_{\mathcal{SP}}, tr_i \in \text{serials}(G_i) : \\
&\qquad tr' \in tr_1 \frown \ldots \frown t_n \wedge tr \leqslant tr' \\
&\Leftrightarrow \quad \forall\, i \in \{1, \ldots, n\}, \exists\, tr_i \in \mathit{traces}(\text{trans}(t_i)) \setminus \checkmark : \\
&\qquad tr' \in tr_1 \frown \ldots \frown t_n \wedge tr \leqslant tr' \text{ by induction hypothesis} \\
&\Leftrightarrow \quad \forall\, i \in \{1, \ldots, n\}, \exists\, tr_i' \leqslant tr_i \in \mathit{traces}(\text{trans}(t_i)) \setminus \checkmark : \\
&\qquad tr \in tr_1' \frown \ldots \frown t_n' \\
&\Leftrightarrow \quad tr \in \mathit{traces}(\text{trans}(t)) \setminus \checkmark.
\end{aligned}
$$

8.4.4 *Implementation*

We provide a prototype implementation of our proposed methodology. This implementation is built using Python 2.7 and requires as input an attack tree and a formal model of the SUT. It then automatically carries out three main tasks:

1. Translates an input (manually pre-determined) attack tree into a CSP process;
2. Uses this process and the formal model of the SUT to generate test cases; and
3. Executes all the generated (scriptable) test cases by associating each one with a sequence of predefined primitive test scripts.

Task 1 is a straightforward implementation of function `trans` from Section 8.4.3. In the rest of this section, we discuss the implementation of tasks 2 and 3 in detail.

8.4.4.1 *Test Case Generation*

Let us assume that the formal model of the SUT is given as a CSP process Sys. Furthermore, the behaviours of the attacker are also given

in terms of an attack tree t, which is then transformed into a CSP process trans(t). We shall use trace refinement in CSP to extract test cases following [117]. To this end, trans(t) acts as a filter criterion to select test cases among all possible runs of the system captured by Sys. As in [117], we define a fresh event *attackSucceed* to mark the end of an attack, which indicates that an attack is successfully executed. We form the following filter:

$$TestPurpose = \text{trans}(t);\ (attackSucceed \rightarrow Stop)$$

which captures all attacks extended with the marking event *attackSucceed* at the end. Then, we establish the following trace refinement:

$$Sys \ \Box\ TestCases \sqsubseteq_T Sys \underset{\Sigma \setminus \{attackSucceed\}}{\parallel} TestPurpose$$

In this refinement, $TestCases$ encodes test cases that have previously been generated. By combining it with Sys using the external choice operator, a fresh test case, i.e., different from the generated ones, will be generated if one exists. $Sys \underset{\Sigma \setminus \{attackSucceed\}}{\parallel} TestPurpose$ encapsulates all attack traces that can be carried out with respect to the formal model Sys. These attack traces are ended with the marking event *attackSucceed*, which does not belong to Sys, which, hence, gives rise to counter examples of the refinement.

Initially, $TestCases = TestCases_0 = Stop$ (i.e. corresponding to an empty set of test cases). This refinement is checked by calling FDR [150]. If an attack trace exists, FDR will provide a counter example of the form $\langle a_1, \ldots, a_n, attackSucceed \rangle$ where $a_1, \ldots, a_n \in \Sigma \setminus \{attackSucceed\}$. We encode this trace as a test case $tc_1 = a_1 \rightarrow \ldots \rightarrow a_n \rightarrow attackSucceed \rightarrow Stop$. After $TestCases$ is rebuilt as $TestCases = TestCases_1 = TestCases_0 \ \Box\ tc_1$, the above refinement check is called again and again to extract further test cases $tc_2, \ldots$ and to construct $TestCase_2, \ldots$ until no further counter example can be found.

In this implementation, the calls to checking refinements and extracting counter examples are facilitated by API functions provided by FDR [150].

8.4.4.2 *Test Case Execution*

Test cases that are generated can now be assigned programmatic functions that would allow for execution. This is dependant on implementation of the system and so would necessarily be specific rather than abstract. Furthermore, as the attack tree in this case is based on penetration testing, not all actions (such as "social engineering") are scriptable, largely due to requiring manual intervention. All such actions are indicated in the implementation.

Given an attack tree t, let $\mathit{scriptable}(t)$ denote the set of its scriptable leaves. Each scriptable leaf $a \in \mathit{scriptable}(t)$ is associated with a primitive test script $\mathit{script}(a)$. A generated test case $tc = a_1 \rightarrow \ldots \rightarrow a_n \rightarrow \mathit{attackSucceed} \rightarrow \mathit{Stop}$ is automatically executable if $a_i \in \mathit{scriptable}(t) \, \forall \, i = 1, \ldots, n$.

Execution of an automatically executable test case means the execution of all test scripts $\mathit{script}(a_1), \ldots, \mathit{script}(a_n)$ sequentially. If all such scripts are executed successfully, the test case is called *passed*, otherwise *failed*. Note that a passed test case means that the SUT is not secure with respect to the attack encoded by this test case. If the converse is true, then the SUT is impervious to this attack.

In this paper, we use the example of an aftermarket on-board diagnostics (OBD-II) dongle attached to the vehicle (this is further discussed in Section 8.4.4). Executable test cases are written in Python 2.7 to enable compatibility with Bluetooth functions on a generic Linux distribution.

8.4.5 *Case Study*

We take here a case study of evaluating the intra-vehicular network with the attack goal of vehicle compromise. The attack tree (see Section 8.4.5.1) for this goal is based around access through a Bluetooth-enabled aftermarket device that attaches to the vehicle's on-board diagnostic (OBD-II) port. Detailed information regarding the OBD-II device and its communications can be found in Chapter 7.

8.4.5.1 *Attack Tree Translation*

The attack tree used for this case study is shown in Figure 23.

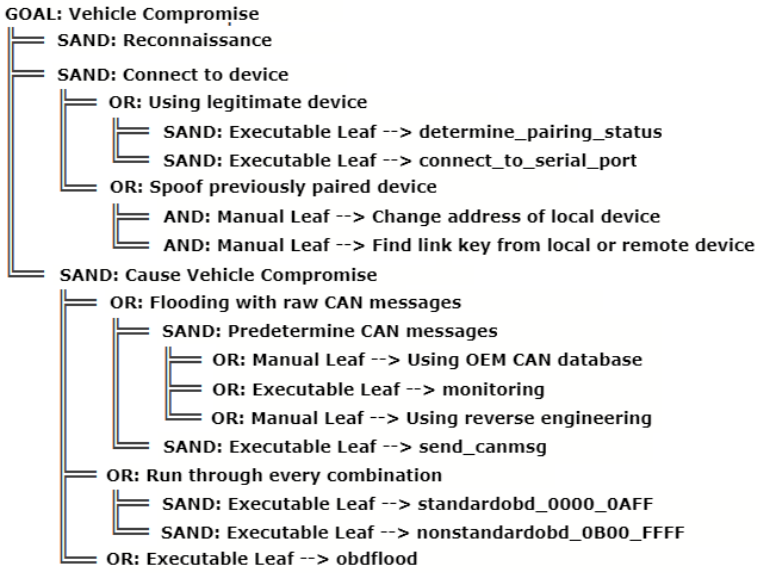

Figure 23: Attack tree, with attack goal of compromising the vehicle through an aftermarket Bluetooth-enabled OBD-II device

Figure 23 also lists test scripts corresponding to leaves in this attack tree. If a leaf is not scriptable, it is denoted as a *manual leaf*.

The function `trans(Vehicle Compromise)` (see Section 8.4.3), gives the translation of this tree into CSP:

channel $action\_Determine\_pairing\_status$

channel $action\_Connect\_to\_serial\_port$

channel $action\_Change\_address\_of\_local\_device$

channel $action\_Find\_the\_link\_key\_from\_local\_or\_remote\_device$

channel $action\_Send\_flood\_with\_CAN\_messages$

channel $action\_Using\_OEM\_CAN\_database$

channel $action\_Using\_passive\_monitoring$

channel $action\_Using\_reverse\_engineering$

channel $action\_Run\_through\_standard$

channel $action\_Run\_through\_non\_standard$

channel $action\_Flood\_with\_set\_OBD\_messages$

$Attacker = Vehicle\_Compromise$

$Vehicle\_Compromise = Connect\_to\_device;\ Cause\_Vehicle\_Compromise$

$Connect\_to\_device = Using\_legitimate\_device \ \square \ Spoof\_previously\_paired\_device$

$Using\_legitimate\_device = Determine\_pairing\_status;\ Connect\_to\_serial\_port$

$Determine\_pairing\_status = action\_Determine\_pairing\_status \rightarrow Skip$

$Connect\_to\_serial\_port = action\_Connect\_to\_serial\_port \rightarrow Skip$

$Spoof\_previously\_paired\_device =$

$\quad Find\_the\_link\_key\_from\_local\_or\_remote\_device$

$$
\begin{aligned}
&\quad ||| \ Change\_address\_of\_local\_device \\
&Change\_address\_of\_local\_device = \\
&\quad action\_Change\_address\_of\_local\_device \rightarrow Skip \\
&Find\_the\_link\_key\_from\_local\_or\_remote\_device = \\
&\quad action\_Find\_the\_link\_key\_from\_local\_or\_remote\_device \rightarrow Skip \\
&Cause\_Vehicle\_Compromise = \\
&\qquad Using\_OBD\_messages \\
&\quad \Box \ Run\_through\_all\_messages \\
&\quad \Box \ Flooding\_with\_raw\_CAN\_messages \\
&Flooding\_with\_raw\_CAN\_messages = \\
&\quad Predetermine\_CAN\_messages; \ Send\_\_flood\_with\_\_CAN\_messages \\
&Predetermine\_CAN\_messages = \\
&\qquad Using\_passive\_monitoring \\
&\quad \Box \ Using\_OEM\_CAN\_database \\
&\quad \Box \ Using\_reverse\_engineering \\
&Send\_flood\_with\_CAN\_messages = action\_Send\_flood\_with\_CAN\_messages \rightarrow \\
&Skip \\
&Using\_OEM\_CAN\_database = action\_Using\_OEM\_CAN\_database \rightarrow Skip \\
&Using\_passive\_monitoring = action\_Using\_passive\_monitoring \rightarrow Skip \\
&Using\_reverse\_engineering = action\_Using\_reverse\_engineering \rightarrow Skip \\
&Run\_through\_all\_messages = Run\_through\_standard; \ Run\_through\_non\_standard \\
&Run\_through\_standard = action\_Run\_through\_standard \rightarrow Skip \\
&Run\_through\_non\_standard = action\_Run\_through\_non\_standard \rightarrow Skip \\
&Using\_OBD\_messages = Flood\_with\_set\_OBD\_messages \\
&Flood\_with\_set\_OBD\_messages = action\_Flood\_with\_set\_OBD\_messages \rightarrow \\
&Skip
\end{aligned}
$$

Reconnaissance was defined to find as much information as possible (meaning the subsequently generated formal attack tree would be much larger). Many of the steps were manual and non-sequential. As such, reconnaissance actions were considered out of scope for this exercise.

8.4.5.2 *Results*

We generate test cases using the implementation in Section 8.4.4.1. Given the small size of the attack tree, we use the most abstract model for the SUT where all behaviours are accepted (the most insecure model), to generate a total of 15 test cases. Results from the run of test cases against an actual implementation are given in Table 31.

Three of the test cases passed (i.e. they were executed successfully):

$$TC(3) = action\_Determine\_pairing\_status \rightarrow$$
$$action\_Connect\_to\_serial\_port \rightarrow$$
$$action\_Flood\_with\_set\_OBD\_messages \rightarrow attack\_succeed \rightarrow Stop$$
$$TC(6) = action\_Determine\_pairing\_status \rightarrow$$
$$action\_Connect\_to\_serial\_port \rightarrow action\_Run\_through\_standard \rightarrow$$
$$action\_Run\_through\_non\_standard \rightarrow attack\_succeed \rightarrow Stop$$
$$TC(8) = action\_Determine\_pairing\_status \rightarrow$$
$$action\_Connect\_to\_serial\_port \rightarrow$$
$$action\_Using\_passive\_monitoring \rightarrow$$
$$action\_Send\_flood\_with\_\_CAN\_messages \rightarrow attack\_succeed \rightarrow Stop$$

Further analysis from this is possible. For example, with test case 3 ($TC(3)$ above), the action of flooding with a particular diagnostic message resulted in loss of function in the vehicle of both electronics and engine (see Chapter 7 for empirical test results). This violates the security property of availability by causing a denial of service. Additionally, injection of messages into the CAN bus also changes the stream of CAN bus signals that would normally be expected in vehicles. This violates the security property of integrity (in which no unauthorised modification should be allowed). Both these properties could be addressed in a future model. The tree could also be run against that model iteratively to check whether sufficient protection has been afforded by the improved design.

Protecting against this could involve the addition of a gateway in the SUT, which could either filter out floods of messages (by defining thresholds for the number of these messages that could be sent through at any given time). Alternatively, such messages could be disallowed by having the gateway only allow messages that have come from an authorised source.

The other test cases (all involving permutations of finding a link key, and changing the address) were not scripted because they required manual intervention. The former because it would need a remote device set to enable logging on the Host Controller Interface (not always possible) or to manually acquire data from a vehicle to find where the link key has been stored (which would have required hardware removal). The latter is automatable (for example, using the tool *Spooftooph* [43]), however, either hardware removal or social manipulation is involved to find knowledge of an address that is already stored on the vehicle.

TC#	Execution result
1	unexecutable action_Find_the_link_key_from_local_or_remote_device
2	unexecutable action_Change_address_of_local_device
3	Passed
4	unexecutable action_Find_the_link_key_from_local_or_remote_device
5	unexecutable action_Change_address_of_local_device
6	Passed
7	unexecutable action_Find_the_link_key_from_local_or_remote_device
8	Passed
9	unexecutable action_Using_OEM_CAN_database
10	unexecutable action action_Using_reverse_engineering
11	unexecutable action_Change_address_of_local_device
12	unexecutable action_Find_the_link_key_from_local_or_remote_device
13	unexecutable action_Change_address_of_local_device
14	unexecutable action_Change_address_of_local_device
15	unexecutable action_Find_the_link_key_from_local_or_remote_device

Table 31: Test cases that were run against a real world vehicle

Other branches that were unscripted involves reverse engineering a CAN message to inject, as this involves manual trial and error currently due to the sheer volume and variety of messages that are on the CAN bus at any one time. Using an OEM's CAN database would enable (partial) automation, but availability is often non-existent due to commercial confidentiality. The branch that ended with successful test cases all involved using a legitimate device. That is, a device that was under our control, which we could use to test the weaknesses in the vehicular implementation.

8.4.6 *Summary*

We have demonstrated the translation of an informal attack tree into a formal process algebra CSP and proved equivalence. We use this tree to generate test cases automatically, and assign executions to scriptable test cases. We then execute the test cases on a real-world vehicle (although this could be substituted with a testbed, with input from an OEM to reflect a real architecture, without the cost or risk to a test vehicle [53]). Thus, the full testing process is one step further

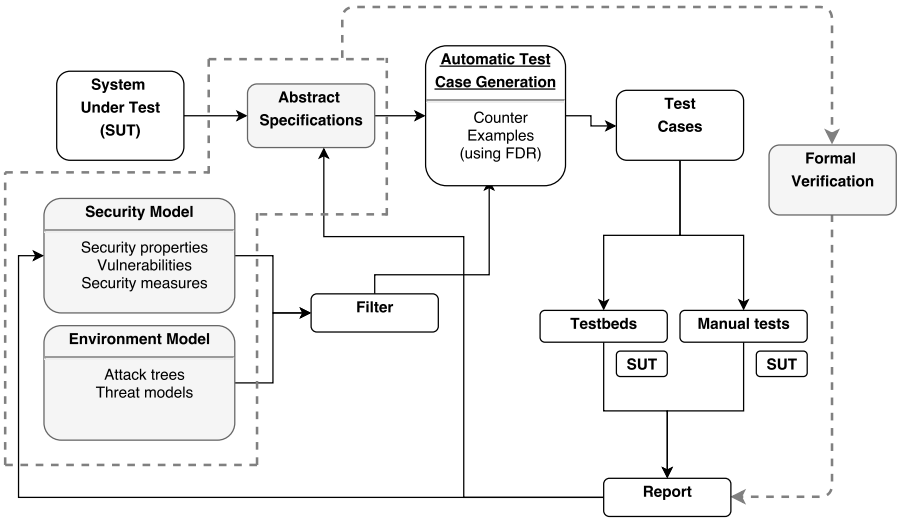

Figure 24: Ideas for further formal analysis

to automation, and furthermore, the formal model of the attack tree could also be used for formal verification should the specifications of the system-under-test be available. Limitations are around how a tree is created (still largely manual) and certain actions within the attack tree requiring manual intervention.

8.4.7 *Future Challenges*

The use of results in formal design and the translated attack trees could potentially fill the chasm between the two streams of testing and design. An extension of the above would require that additional models be first created:

- a model using abstract specifications of the system under test;
- a security model (informed by potential security measures and properties that we want to test against) and
- finally, an environment model informed by attack trees (as seen above) and threat models.

These models could be built using a formal language (such as the process algebra CSP) and, with appropriate filtering (taking into account of components and security properties to be tested), could then be used to generate the appropriate test cases. We would then follow through using normal model based testing processes. If necessary, this can be performed concurrently with manual testing. Reports that

are outputted from these tests (whether formal or manual) could then be used to refine any of the three models described above.

The entire process would (at a very abstract level) thus appear as in Figure 24, incorporating the work from earlier on in this chapter and, if necessary, the empirical evaluation as performed in earlier chapters.

The work with testbeds (as in work done in [53]) as well as manual testing could also continue to assist in further refinement of the models created.

Once practicalities have been determined, the methodology could then be used in-house as part of an already existing process and in accordance with an industry standard of choice.

9

CONCLUSIONS AND SUMMARY OF CONTRIBUTIONS

In this chapter, the conclusions drawn from the research above are presented (Section 9.1) followed by a summary of contributions (Section 9.2). Finally, ideas for future direction and further work on all aspects of the work in this thesis are outlined in Section 9.3.

9.1 CONCLUSIONS

The threat landscape of the vehicle is evolving, primarily driven by an increase in both functionality and connectivity. This has opened a closed system to external influence, compounded complexity and made it difficult to audit for security flaws and weaknesses. Many studies have shown that the vehicle is fundamentally insecure, and that systematic security testing is essential.

Bluetooth was taken as an example interface because of its ubiquity, and because many Bluetooth-enabled vehicles and attached aftermarket devices on the road could potentially be accessible to attackers.

A systematic approach is advantageous in that coverage is greater. There are many related approaches, both formal and informal. However, specifications, source code or any detailed technical information is scarce. This precludes many formal model-based testing methods and leads to the evaluation as performed in this research: systematic black box penetration testing supported by attack trees.

This was implemented as a semi-automated proof-of-concept tool and tested on the native Bluetooth interface of eight vehicles, as well as Bluetooth-enabled aftermarket devices attached to the vehicle's on-board diagnostics port. The systematic approach found many weaknesses, based on the attack goals of data extraction, denial of service and, more generally for the aftermarket devices, vehicle compromise.

As attack trees systematise and document both tests and reactions, the results of such testing could be used as evidence in security assurance cases, or in future formal methods exercises. The latter could be achieved by inferring requirements and generating additional speci-

fications, which can be cross-referenced with other requirements and traced back through the pre-determined attack trees. Alternatively, model-based testing could be made possible through the translation of informal attack trees into formal structures. Thus formal testing could be performed, albeit with its correctness dependent on the availability of trustworthy system-under-test specifications.

We envision that end users such as security analysts and engineers would be able to use this framework to assure and enhance the cybersecurity of vehicles.

9.2 SUMMARY OF CONTRIBUTIONS

This doctoral research has laid the foundations for an automated systematic evaluation of an automotive interface. Summarised here are the contributions addressing the research questions as laid out in Section 1.2:

- The motivation for investigating vehicles from a security perspective, as well as the situational awareness regarding the state of Bluetooth security in vehicles is addressed in Chapter 3.4, with the contribution being a threat intelligence study detailing both the technological lag found in vehicles as well as the number of vehicles and aftermarket devices that could be accessed on the road even with a low powered device;

- Addressing the question of how to establish a baseline security state, without technical specifications is the main contribution of the thesis (detailed in Chapter 4): towards a systematic security evaluation of the automotive Bluetooth interface (whether that interface be native, or enabled through an aftermarket device). This evaluation is implemented as a proof-of-concept tool (Chapter 5), which also contributes towards the automation of the evaluation;

- The results of testing can be used to aid in security assurance cases (described in Chapter 6 and 7), as well as be used in future formal model-based testing or further iterations of design (Chapter 8).

9.3 FUTURE WORK

Along with the concepts presented in previous chapters, the research as a whole in this thesis could also be further improved in the short-term by:

- Extending on the number of test vehicles, and drawing more comparisons between them, both with the severity classification and the aftermarket devices. This would depend on the availability of funding and facilities;
- Expanding the existing attack trees to include other attacks, involving jamming, tracking through signal strength and incorporating output from a Bluetooth sniffer;
- Defining new attack goals (such as attacking privacy) and building the appropriate attack trees (and supporting tool chain); and
- Extending the tool to other wireless technologies such as WiFi, and performing the same systematic testing in order to enumerate security issues through other vectors.

The project also opens up several new (longer term) avenues of research. The use of the severity rating process could be used as part of the risk management process as described in J3061. Other studies such as [158] have already looked at using similar methods in a risk analysis process, and our methods could integrate with risk analyses such as these to form a whole management process.

This would necessitate automation of some of the prioritisation process, where the aspects with zero ratings and very high S4 ratings could be automatically classified. Automation would also include the calculation of priority of everything else in the grey areas. Information regarding the type of component being tested and the priority of the tester should be taken into account during these calculations.

Some aspect of safety analysis and risk to financial transactions in a vehicle (assuming the availability of a car with this technology deployed) could also be included in future classifications to more accurately reflect what is stated in EVITA. This would also provide further granularity in terms of the evidence attached to a security assurance case.

Deployment of wireless technology and subsequently connectivity is only increasing, both in the number and variety of interfaces and

protocols used as well as bandwidth capacities. Going forward to where vehicles become "smarter" and eventually autonomous [34], the need to evaluate the security of wireless connections will never be less than essential.

ANNOTATED BIBLIOGRAPHY

The more that you read,
the more things you will know.
The more that you learn,
the more places you'll go.

— Dr. Seuss

Presented below is an annotated bibliography. The annotations were drawn from notes and summaries written about each paper as a memory aid. They were particularly helpful in the case of exploits against and tools for working with Bluetooth, since they generally used a variation of the word "Blue" and [insert word]. On a more serious note, these annotations were incredibly useful throughout the course of my programme and are included here for easy perusal.

[1] 115th Congress, *S.680 - SPY Car Act of 2017*, 2017. [Online]. Available: https://www.congress.gov/bill/115th-congress/senate-bill/680 (visited on 2017-10-20),

> *New proposed legislation regarding security and privacy in the vehicle, advocating and proposing mandatory steps taken to protect vehicles against malicious attacks. It also proposes the use of cyber dashboards for consumers so that they are informed of the cybersecurity measures available in the vehicle.*

[2] "A Risk Assessment Framework for Automotive Embedded Systems," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, Xi'an: ACM New York, May 2016, pp. 3–14,

> *Paper describing the HEAVENS project, the use of STRIDE and how it fits within the various phases of automotive production, especially in relation to ISO26262*

[3] A. Aleksandrov, *Bluetool*, 2017. [Online]. Available: https://github.com/emlid/bluetool (visited on 2017-05-05),

> *Source for a Python API for Bluetooth d-bus calls, used in the proof-of-concept tool*

[4] J. Alfaiate and J. Fonseca, "Bluetooth security analysis for mobile phones," in *Proceedings of the 7th Iberian Conference on Information Systems and Technologies (CISTI)*, Madrid, Spain: IEEE, Jun. 2012, pp. 1–6,

An article summarising, describing and exploring Bluetooth security in mobile phones. It contains information about many of the common exploits and techniques used to compromise Bluetooth-enabled mobile phones

[5] Amenaza Technologies Limited, *Fundamentals of Capabilities-based Attack Tree Analysis*, Calgary, 2005. [Online]. Available: http://www.amenaza.com/downloads/docs/AttackTreeFundamentals.pdf (visited on 2014-10-13),

This company creates commercial software for building attack trees, and this page describes some useful concepts of attack trees.

[6] K. Arai and T. Kaneko, "Formal Verification of Improved Numeric Comparison Protocol for Secure Simple Paring in Bluetooth Using ProVerif," in *Proceedings of the 2014 International Conference on Security and Management (SAM) at the Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, Las Vegas: WorldComp, Jul. 2014,

Looks at formal verification of Secure Simple Pairing (specifically the numeric comparison association model) using a well known tool called ProVerif. They show that it is susceptible to attacks, propose countermeasures and verify its security (using secrecy as a property).

[7] Argus, *Argus Cyber Security Working With Bosch to Promote Public Safety and Mitigate Car Hacking*, 2017. [Online]. Available: https://argus-sec.com/argus-cyber-security-working-bosch-promote-public-safety-mitigate-car-hacking (visited on 2017-04-26),

Presents a demonstration by a commercial company interested in cybersecurity of compromise of the SSP mechanism (Just Works) model, as well as demonstration of a compromise of vehicle through an attached OBD-II device on the vehicle

[8] Auto ISAC, *Automotive Information Sharing and Analysis Centre*, 2016. [Online]. Available: https://www.automotiveisac.com/best-practices/ (visited on 2017-05-12),

An organisation promoting and enhancing knowledge sharing, specifically on cybersecurity issues, in the automotive industry. However, despite its promotion of it, its regulations still stipulate anonymity of data and origin unless it is absolutely necessary for it to be otherwise.

[9] J. Barnickel, J. Wang, and U. Meyer, "Implementing an attack on Bluetooth 2.1+ secure simple pairing in Passkey Entry mode," in *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, Jun. 2012, pp. 17–24,

One of the many sources used when researching possible attacks on Bluetooth. The attack involves eavesdropping on an ongoing pairing process, recording all message exchanges. The session is then interrupted through jamming, with the attacker now able to calculate the passkey used in the interrupted session. The calculation of link keys belonging to the legitimate devices attempting to pair is dependent on the PIN being reused when the jamming is stopped. If successful, the attacker is then a man-in-the-middle and able to eavesdrop on communications.

[10] S. Bayer, T. Enderle, D. K. Oka, and M. Wolf, "Security Crash Test – Practical Security Evaluations of Automotive Onboard IT Components," in *Electronic Proceedings of the 2014 ESCRYPT Automotive Safety and Security Conference*, Stuttgart: ETAS, Apr. 2015,

Proposes a set of automotive security evaluation assurance levels which describe what tests have been performed in order to provide assurance.

[11] Benhui.net, *Bluetooth (jabwt) browser midlet*, 2003. [Online]. Available: http://www.benhui.net/bluetooth/btbrowser.html,

Old Bluetooth tool used to gather information from surrounding Bluetooth devices. Works on phones that support Java Bluetooth (targeted specifically at Nokia 6600)

[12] A. Bertolino, "Software Testing Research : Achievements , Challenges , Dreams," in *Proceedings of the 2007 Conference on Future of Software Engineering (FOSE)*, Minneapolis: IEEE Computer Society, May 2007, pp. 85–103,

Presents a state-of-the-art regarding software testing research. Not specifically dedicated to security, but with useful concepts surrounding the challenges in software testing, including hypotheses, effectiveness and empiricism.

[13] M. Bishop, "What is computer security?" *Security & Privacy, IEEE*, vol. 1, no. 1, pp. 67–69, Jan. 2003,

Explores at a high level what is meant by security requirements, policy, mechanisms and assurance

[14] M. Bishop and D. Bailey, "A Critical Analysis of Vulnerability Taxonomies," University of California, Tech. Rep. September Issue, 1996. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA453251 (visited on 2017-05-23),

Technical report looking at the classification of vulnerabilities into a set of tuples, aiming to provide a assistance in detecting new vulnerabilities and collating classes of vulnerabilities. Distinguishes between attack and vulnerability

[15] BlueZ Project, *BlueZ*, 2017. [Online]. Available: http://wwww.bluez.org/ (visited on 2017-05-10),

The official Linux Bluetooth stack

[16] K. Blueriver, *PTable*, 2016. [Online]. Available: https://pypi.python.org/pypi/PTable/0.9.0 (visited on 2017-03-14),

Tool used for creating and displaying ASCII tables

[17] Bluetooth Penetration Testing Framework, *Bluetooth Penetration Testing Framework*, 2011. [Online]. Available: http://bluetooth-pentest.narod.ru/ (visited on 2016-08-11),

This is more of a collection of links to useful Bluetooth investigation snippets and code examples, rather than an official framework. There was no information on this site as to who or when these snippets were put together.

[18] Bluetooth SIG Inc., *Personal Area Network (PAN)*, 2003. [Online]. Available: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=6554 (visited on 2016-12-16),

Details and specifications for the Bluetooth Personal Area Network service profile

[19] ——, *Bluetooth SIG: Core Version 2.1+EDR*, 2007. [Online]. Available: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=241363 (visited on 2017-02-06),

Specifications for Bluetooth version 2.1+EDR

[20] ——, *Bluetooth profiles overview*, 2015. [Online]. Available: https://developer.bluetooth.org/TechnologyOverview/Pages/Profiles.aspx (visited on 2017-04-13),

A list of profiles and services supported by the current version of Bluetooth

[21] ——, *File Transfer Profile (FTP) 1.3.1*, 2015. [Online]. Available: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=309003 (visited on 2017-04-05),

Details and specifications for the Bluetooth File Transfer Profile service

[22] ——, *Bluetooth SIG: Deprecated Specifications*, 2016. [Online]. Available: https://www.bluetooth.com/specifications/adopted-specifications/deprecated-specifications (visited on 2017-06-02),

A list of deprecated specifications for Bluetooth

[23] ——, *Bluetooth SIG: Legacy Specifications*, 2016. [Online]. Available: https://www.bluetooth.com/specifications/adopted-specifications/legacy-specifications (visited on 2017-06-02),

A list of legacy specifications for Bluetooth

[24] ——, *Our History*, 2016. [Online]. Available: https://www.bluetooth.com/about-us/our-history (visited on 2017-02-06),

Presents information about the history of Bluetooth, version adoption years and features that were added to new versions.

[25] D. Boddie, *PyOBEX Python Package*, 2015. [Online]. Available: http://www.boddie.org.uk/david/Projects/Python/PyOBEX/ (visited on 2016-10-10),

A Python wrapper for the Bluetooth OBEX functionality

[26] W. Bronzi, T. Derrmann, G. Castignani, and T. Engel, "Towards Characterizing Bluetooth Discovery in a Vehicular Context," in *in Proceedings of the 2016 IEEE Vehicular Networking Conference (VNC)*, Columbus, Ohio: IEEE, 2016,

A large scale study conducted by wardriving, looking to characterise any Bluetooth-enabled device that might be found on a typical drive. Concentrated geographically in Luxembourg. Security is only briefly mentioned towards the end, as the emphasis was on looking at Bluetooth as a possible V2V and V2I communication protocol, but is useful as a basis for any other wardriving exercises.

[27] P. J. Brooke and R. F. Paige, "Fault trees for security system design and analysis," *Computers & Security*, vol. 22, no. 3, pp. 256–264, 2003,

Fault trees are usually used for functional safety purposes. Describes the building and analysis of a fault tree, and proposes a use for fault trees in security and states that fault trees could be used to explicitly identify relationships between events as well as a realistic framework in order to build a security assurance case.

[28] A. Buldas, P. Laud, and J. Priisalu, "Rational choice of security measures via multi-parameter attack trees," *Critical Information Infrastructures Security*, vol. 4347, no. 1, pp. 235–248, 2006,

Aimed at improving the security of institutions through a risk analysis based method involving attack trees

[29] P. Burnley, *Electronic Theft Tool Identification Guide*, Ryton: ACPO Vehicle Crime Intelligence Service, 2014,

Presentation regarding devices and tools that can be used to steal a car and are commonly in use today.

[30] E. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," in *Proceedings of the 2004 International Infrastructure Survivability Workshop (IISW'04)*, Lisbon: IEEE, 2004,

Applies attack trees to the problem of security in industrial control systems, specifically looking at a common SCADA protocol called MODBUS. Sample trees are shown, and validated by domain expert review.

[31] E. Chai, B. Deardorff, and C. Wu, "Hacking Bluetooth," 2012, [Online]. Available: https://css.csail.mit.edu/6.858/2012/projects/echai-bendorff-cathywu.pdf (visited on 2015-05-21),

Working paper containing an outline of the workings of the Bluetooth protocol, as well as detailing some of the more common attacks against this protocol

[32] R. Chang and V. Shmatikov, "Formal analysis of authentication in bluetooth device pairing," in *Proceedings of the 2007 Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA)*, Wroclaw, Poland, 2007, p. 45,

Presents a formal analysis of the standard Bluetooth pairing protocol using ProVerif, and confirm a previously described guessing attack. Particularly looking at numeric comparison and out-of-band authentication. They also incorporate session identifiers such that the authentication properties still hold in the new model. The ultimate aim is towards automated formal analysis authentication protocols that involve humans.

[33] M. Cheah, S. A. Shaikh, O. Haas, and A. Ruddle, "Towards a systematic security evaluation of the automotive Bluetooth interface," *Journal of Vehicular Communications*, vol. 9, no. 2017, pp. 8–18, 2017,

Describes, implements and demonstrates the systematic security evaluation of the automotive Bluetooth interface, using attack trees. A proof-of-concept tool is also present, contributing towards the automation of the process, in the context of the automotive native Bluetooth interface

[34] M. Cheah and S. Shaikh, "Autonomous Vehicle Security," *IET Engineering and Technology Reference*, vol. 1, no. 1, 2015,

Review paper regarding the positions around what could affect the security of autonomous vehicles

[35] M. Cheah, S. Shaikh, J. Bryans, and H. N. Nguyen, "Combining third party components securely in automotive manufacture," in *Proceedings of 10th International Conference on Information Security Theory and Practice*, Crete: Springer, 2016,

Conceptual methodology allowing for the use of results from security testing (using attack trees and penetration testing) in a future formal design method.

[36] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces.," in *Proceedings of 20th USENIX Security Symposium*, San Francisco, CA: USENIX Association, 2011, pp. 77–92,

The authors perform experimental analyses around a variety of external short range and long range wireless. Work based around physical access to a car had already been addressed in earlier work. Theory to understand the problem is primarily around threat modelling and looking at related work such as that done with the TPMS or RFID related attack. They proceed to describe successful attacks through Bluetooth, cellular, CD player, radio and OBD-II

[37] X. Chen, *Treelib documentation (revision bd53bbdf)*, 2014. [Online]. Available: http://treelib.readthedocs.io/en/latest/ (visited on 2016-11-19),

Tool for building, displaying and searching trees in Python

[38] K.-T. Cho and K. G. Shin, "Error Handling of In-vehicle Networks Makes Them Vulnerable," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, Vienna: ACM New York, NY, Oct. 2016, pp. 1044–1055,

Paper describing how to turn error-handling functionality against the CAN network, by inducing errors (by injecting invalid packets). This results in the transmission error counters being increased, which results in a mistakenly categorised "defective" ECU, which triggers the CAN fault confinement to force victim ECU(s) to shut down (i.e. bus-off state).

[39] K. Daley, R. Larson, and J. Dawkins, "A structural framework for modeling multi-stage network attacks," in *Proceedings. International Conference on Parallel Processing Workshop*, IEEE Computer Society, 2002, pp. 5–10,

Uses attack trees with an enhancement called Stratified Node Topology, which looks at the levels of the attack tree and assigns hierarchical levels (event level, state level and top level). The top level seems synonymous with attack goal, with the state level being the attack pattern and the event level the attack method. A very useful way of looking at how to break down the trees,

and was used to create the initial "map of the universe" attack tree before the focus on Bluetooth.

[40] A. Dardanelli, F. Maggi, M. Tanelli, S. Zanero, S. M. Savaresi, R. Kochanek, and T. Holz, "A security layer for smartphone-to-vehicle communication over bluetooth," *IEEE Embedded Systems Letters*, vol. 5, no. 3, pp. 34–37, 2013,

The contribution is a control system architecture which integrates a phone with an embedded system (in order for the phone to interact securely with a modern vehicle without requiring any modifications). This is to address the problem of the extension of system boundaries over which a manufacturer has no control.

[41] J. Dawkins and J. Hale, "A systematic approach to multi-stage network attack analysis," in *Proceedings of the 2nd IEEE International Information Assurance Workshop*, 2004, pp. 48–56,

Presents a framework for network attack modeling and analysis, including modelling vulnerabilities and attacker capabilities. A prototype is also presented which implements the framework, supporting generation of attack tree and correlative analysis of network vulnerabilities. Of interest here are the use of attack trees in the use of various (potentially related) vulnerabilities, although the context is very different.

[42] Department for Transport, "The Pathway to Driverless Cars : A detailed review of regulations for automated vehicle technologies," Department for Transport, Tech. Rep., Feb. 2015. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/401565/pathway-driverless-cars-main.pdf (visited on 2017-06-05),

Technical report collating responses from the automotive and insurance industries, legal professionals, technical institutions and road users regarding attitudes and current regulations surrounding testing of autonomous and highly automated vehicles. Legislation in many other countries apparently prevents such testing, which might allow the UK to fill the gap.

[43] J. Dunning, *Spooftooph*, 2012. [Online]. Available: http://tools.kali.org/wireless-attacks/spooftooph (visited on 2016-09-12),

Proof of concept tool used to change the Bluetooth address, name and class of local HCI adaptor. The changing of the Bluetooth address can only be performed on CSR chips. The latter is similar to what the hcitool suite can do.

[44] J. P. Dunning, "Taming the blue beast: A survey of bluetooth based threats," *IEEE Security and Privacy*, vol. 8, no. 2, pp. 20–27, 2010,

As Bluetooth finds its way into millions of devices worldwide, it also becomes a prime target for hackers. The author presents a taxonomy for threats against Bluetooth-enabled devices, describes several of these threats, and identifies steps for threat mitigation.

[45] ELM Electronics, *ELM Electronics: OBD*, n.d. [Online]. Available: https://www.elmelectronics.com/products/ics/obd/ (visited on 2017-12-04),

Company who manufacture the ELM327 chipsets that are integrated in Bluetooth OBD-II dongles

[46] K. S. Edge, "A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees," PhD Thesis, Air University, 2007. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA472310 (visited on 2015-03-15),

A thesis exploring vulnerabilities in complex systems using attack and protection (countermeasure) trees. Good explanations regarding attack trees and how they can be used, and a detailed exploration of attack trees is available.

[47] S. Edwards, "A framework for practical, automated black-box testing of component-based software," *Software Testing, Verification and Reliability*, vol. 11, pp. 97–111, 2001,

Presents a conceptual framework for the automation of black-box testing. Also explores state-of-the-art regarding testing research and concludes that automated black-box testing is feasible.

[48] S Eichler, "A security architecture concept for vehicular network nodes," in *Proceedings of 6th International Conference on Information, Communications and Signal Processing*, Singapore: IEEE, 2007, pp. 1–5,

The research presents security requirements in a vehicular network (pertaining to VANETs) and how to setup and implement a security architecture in a vehicular network node. They take into account decentralised services and centralised telematics services. They also discuss how this would affect (protect) privacy.

[49] M Enev, A Takakuwa, K Koscher, and T Kohno, "Automobile Driver Fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 34–50, 2016,

Performs experiments with regards to profiling and identifying individual drivers of a vehicle based on usage profile drawn from sensors (and CAN data) that is available from a vehicle. They found that it is possible to identify with one hundred percent accuracy, using ninety percent of data from each person. They need only eight minutes of training data to do this. Future work will include driver identification across multiple days worth of data.

[50] European Commission, *eCall in all new cars from April 2018*, 2015. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/ecall-all-new-cars-april-2018 (visited on 2017-05-20),

A project that aims to have ecall technology (where in an emergency, the vehicle dials 112) in all vehicles. This press release states that European Parliament has voted in favour of having this take effect in all new cars from April 2018

[51] M. Felderer, P. Zech, R. Breu, M. Buchler, and A. Pretschner, "Model-based security testing: a taxonomy and systematic classification," *Software Testing Verification and Reliability*, vol. 26, no. 2, pp. 119–148, 2015,

Presents an approach which incorporates analysis techniques that improve vector identification and detects successful attacks against web applications. This allows for more thorough penetration testing, with the discoery of more vulnerabilities.

[52] E. Fernandez, J. Pelaez, and M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool," in *IFIP International Federation for Information Processing*, vol. 242, Orlando,FL: Springer NY, 2007, pp. 345–357,

A pattern is a solution definition to a problem in any given context, which can be used for design and evaluation. This paper flips the concept and instead conceptualises the attack pattern, describing how the attack is performed, its target, location, paths and anything else that might help define the problem. They present a case study of attacks on VoIP networks and show how such a formalism can be useful.

[53] D. S. Fowler, M. Cheah, S. A. Shaikh, and J. Bryans, "Towards a testbed for automotive cyberecurity," in *Process of the 10th International Conference on Software Testing, Verification and Validation: Industry Track (ICST)*, Tokyo, Japan: IEEE, Mar. 2017,

A paper that proposes a testbed architecture and design (at a high level) that uses industry standard software and mimics the reaction that you would get from a vehicle if you were fuzzing or injecting CAN messages through the OBD-II port.

[54] A. Francillon, B. Danev, and S. Capkun, "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars," in *Proceedings of 2011 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2011,

Demonstrates relay attacks on Passive Keyless Entry and Start (PKES) enabled vehicles using wireless physical-layer relays. They show that this attack would work regardless of protocol employed and make or manufacturer of the vehicle.

[55] A. Fuchs, S. Gürgens, G. Pedroza, and L. Apvrille, "EVITA Project - Deliverable D3.4.4: On-Board Architecture and Protocols Attack Analysis," Fraunhofer SIT, Tech. Rep., 2008. [Online]. Available: http://www.evita-project.org/Deliverables/EVITAD3.4.4.pdf (visited on 2017-09-04),

Deliverable describing and discussing attack analysis on vehicular architecture

[56] A. Fuchs and R. Rieke, "Identification of Security Requirements in Systems of Systems by Functional Security Analysis," *Architecting Dependable Systems VII*, vol. 6420, pp. 74–96, 2010,

Looks at gathering security requirements through a process of systematic deduction as as part of the security engineering pro-

cess from a "system-of-systems" perspective using formal methods.

[57] GSMA, "Connected Car Forecast: Global Connected Car Market to Grow Threefold within Five Years," GSMA, Tech. Rep., 2013. [Online]. Available: http://www.gsma.com/connectedliving/wp-content/uploads/2013/06/cl_ma_forecast_06_13.pdf (visited on 2016-12-22),

A report forecasting the state of connectivity in the automotive industry over five years from 2013. Also contains forecasts of Bluetooth capabilities in vehicles.

[58] C. Gamble and K. Pierce, "Design Space Exploration for Embedded Systems Using Co-simulation," in *Collaborative Design for Embedded Systems*, J. Fitzgerald, P. G. Larsen, and M. Verhoef, Eds., Springer-Verlag Berlin Heidelberg, 2014, ch. 10, pp. 199–222,

Use of Automated Co-model analysis and the Crescendo tool in order to explore all possible variations of a design to provide a surface which an engineer can use to select an optimal design

[59] D. Geer and J. Harthorne, "Penetration testing: a duet," in *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas, NV: IEEE Computer Society, 2002, pp. 185–195,

Explores the concept of penetration testing. It is more a science than an art, as there are no falsifiable hypotheses. This is because the complete set of potential insecurities is unknowable and unenumerable; no penetration tester can prove security.

[60] M. Gegick and L. Williams, "Matching attack patterns to security vulnerabilities in software-intensive system designs," in *Proceedings of the 2005 Workshop on Software Engineering for Secure Systems (SESS) — Building Trustworthy Applications*, vol. 30, St. Louis: ACM New York, 2005, p. 1,

Matching of attack patterns to actual vulnerabilities in the design phase, which means that security efforts can start early, and can be integrated into the software design process.

[61] J. D. Gilsinn and R. Schierholz, "Security Assurance Levels : A Vector Approach to Describing Security Requirements," National Institute of Standards and Technology, Tech. Rep., 2010, pp. 1–13. [Online]. Available: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906330 (visited on 2016-04-12),

Comparing safety to security and how safety processes (such as classifying risks into integrity levels) could be analogous to security processes that could be used. Presents a concept called Security Assurance Levels to classify the protection that a system might require.

[62] D. Gordon, T. Stehney, N. Wattas, and E. Yu, "System Quality Requirements Engineering (SQUARE) Methodology: Case Study on Asset Management System," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Tech. Rep. May, 2005. [Online]. Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1464&context=sei (visited on 2016-09-12),

Presents a case study (asset management) using the SQUARE methodology

[63] S. Gürgens, C. Rudolph, A. Mana, and S. Nadjm-Tehrani, "Security Engineering for Embedded Systems - the SecFutur vision," in *Proceedings of the 2010 International Workshop on Security and Dependability for Resource Constrained Embedded Systems*, B. Hamid, C. Rudolph, and C. Rulan, Eds., ACM, Vienna, Austria: ACM New York, NY, 2010, p. 7,

A position paper regarding future vision and engagement in the SeFutur project, specifically focusing on security engineering for embedded systems

[64] K. Haataja and K. Hypponen, "Man-in-the-middle attacks on Bluetooth: a comparative analysis, a novel attack, and countermeasures," in *Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing*, St. Julians: IEEE, 2008, pp. 12–14,

Describes MITM attacks on the Bluetooth protocol, and presents a novel MITM attack on a printer that uses the SSP mechanism.

[65] K. Haataja, "Security threats and countermeasures in Bluetooth-enabled systems," PhD Thesis, University of Kuopio, 2009. [Online]. Available: http://epublications.uef.fi/pub/urn_isbn_978-951-27-0111-7/urn_isbn_978-951-27-0111-7.pdf (visited on 2014-10-19),

PhD thesis regarding Bluetooth threats, attacks and countermeasures

[66] K. Haataja, K. Hypponen, and P. Toivanen, "Ten Years of Bluetooth Security Attacks: Lessons Learned," in *Proceeding of the Reports and Studies in Forestry and Natural Sciences Mini- Conference*, M. Penttonen, Ed., Finland, 2011,

Review paper around state of the art (at time of writing) on Bluetooth security

[67] P. Handel, I. Skog, J. Wahlstrom, F. Bonawiede, R. Welch, J. Ohlsson, and M. Ohlsson, "Insurance Telematics: Opportunities and Challenges with the Smartphone Solution," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 57–70, 2014,

Article discussing the challenges of gathering data from a smartphone versus when information comes directly from a vehicle such as through the OBD port (specifically with regards to insurance telematics).

[68] A. Heinz, *Bug no. 525220: Bluetooth problem caused by fake dongles?* 2011. [Online]. Available: https://groups.google.com/forum/#!topic/linux.debian.bugs.dist/I1PNLTeWRs (visited on 2017-05-01),

Bug report on possible counterfeit CSR dongles with the same Bluetooth MAC address

[69] C. Hoare, *Communicating Sequential Processes*, Electronic. UK: Prentice Hall International, 1985,

Describes the fundamentals and principles of CSP

[70] J. Hoffman, "Simulated Penetration Testing : From " Dijkstra " to " Turing Test ++ "," in *Proceedings of the 25th International Conference on Automated Planning and Scheduling*, Jerusalem: AAAI Press, 2015,

Explores the difficulty of simulating penetration testing, which is broadly taken as simulating a human hacker. The primary difficulty is around sequential decision making, and this study derives a systematic view of the model space to highlight challenges in AI and in sequential decision making research

[71] P. Hope, G. Mcgraw, and A. I. Anton, "Misuse and Abuse Cases: Getting Past the Positive," *IEEE Security & Privacy*, vol. 2, no. 3, pp. 90–92, 2004,

Position paper on how the word "secure" might be defined, especially when security is not a set of features

[72] T. Hoppe, S. Kiltz, and J. Dittmann, "Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats," in *Proceedings of the 28th International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Hamburg: Springer-Verlag Berlin, 2009, pp. 145–158,

Concentrating on privacy threats, this paper describes the challenges of automotive security when IT technologies are introduced. Some attacks are described from a black box perspective, with a case study of reading out data from the vehicular diagnostic port (and speculate that in future, wireless and remote versions of this are possible. They conclude that attackers have common starting points, and that manufacturers should begin to address these issues.

[73] ——, "Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11–25, 2011,

Describes security threats to automotive CAN and demonstrates with practical examples using suppression of warning lights, and undemanded window activity amongst others. Recommendations are then made for the short term to address these threats

[74] Horiba MIRA Ltd., *ISO26262: Update on development of standard*, 2016. [Online]. Available: https://nmi.org.uk/wp-content/uploads/2016/01/HORIBA-MIRA-ISO-26262-NMI-Jan-16-DDW.pdf (visited on 2017-12-05),

A presentation explaining planned updates to the development of the ISO26262 (Functional Safety for Road Vehicles) standard.

[75] J. Hubaux, S Capkun, and J Luo, "The security and privacy of smart vehicles," *IEEE Security and Privacy Magazine*, vol. 2, no. 3, pp. 49–55, 2004,

Review paper around what could happen when smart vehicles start being more intelligent. Old paper

[76] K. Hypponen and K. Haataja, ""Nino" Man-In-The-Middle Attack on Bluetooth Secure Simple Pairing," in *3rd IEEE/IFIP International Conference in Central Asia on Internet*, vol. I, 2007, pp. 1–5,

Describes a novel attack based on falsifying the input and output capabilities during the capabilities exchange phase when two devices connect. This is so that devices can be forced into using Just Works, which has very weak security

[77] IEEE, *IEEE Standards Association: Registration Authority*, 2017. [Online]. Available: https://regauth.standards.ieee.org/standards-ra-web/pub/view.html\#registries,

A database containing registered manufacturers for certain MAC addresses. Contains organisationally unique identifiers for companies that manufacture Bluetooth devices

[78] IHS Markit, *Automotive Wireless Market to Expand by More Than 40 Percent from 2012 to 2018*, 2013. [Online]. Available: http://news.ihsmarkit.com/press-release/design-supply-chain/automotive-wireless-market-expand-more-40-percent-2012-2018 (visited on 2017-06-03),

Executive summary of report regarding wireless connections in automotive and the state of market revenue until 2018.

[79] ISO, "ISO26262-1:2011 Road vehicles - Functional Safety - Part 1: Vocabulary," ISO, Standard ISO TC-22-SC3, 2011,

First part to the automotive functional safety standards

[80] ISO, "ISO 14229-1: Road Vehicles - Unified Diagnostic Services," Standard ISO/TC 22/SC 31, 2013,

ISO standard dealing with the set of diagnostic messages in vehicles called Unified Diagnostic Services.

[81] M. S. Idrees, Y. Roudier, and L. Apvrille, "A framework towards the efficient identification and modeling of security requirements," in *Proceedings of the 5th conference on Network Architecture and Information Systems*, Menton, May 2010, pp. 1–15,

Targets formal modelling of security requirements at early design stages. "Views" are constructed based on functionality and architecture using UML. Used and refined and applied in the context of EVITA. Implemented in a tool called Ttool.

[82] M. Jakobsson and S. Wetzel, "Security Weaknesses in Bluetooth," in *Proceedings of the Cryptographers' Track at RSA Conference*, vol. 1, San Francisco, CA: Springer Berlin Heidelberg, 2001, pp. 176–191,

Early work looking at Bluetooth version 1, and the weaknesses and vulnerabilities that it had. Recommends some basic countermeasures such as physical protection, protecting link keys, extending PIN length and more.

[83] R. Jhawar, B. Kordy, S. Mauw, S. Radomirović, and R. Trujillo-Rasua, "Attack Trees with Sequential Conjunction," in *Proceedings of the 30th IFIP TC 11 International Conference*, vol. 455, Hamburg, Germany, 2015, pp. 339–353,

Provides formal foundations and description of the sequential conjunction operator in attack trees. Give semantics to SAND attack trees, which is used in this thesis (after proof of equivalence) to translate an informal attack tree to a formal attack tree

[84] C. Jones, *PyFiglet*, 2012. [Online]. Available: `https://pypi.python.org/pypi/pyfiglet` (visited on 2017-04-15),

Python tool that creates ASCII art

[85] A Jürgenson and J Willemson, "Processing multi-parameter attack trees with estimated parameter values," in *Proceedings of the 2nd International Conferences on Advance in Information and Computer Security (IWSEC'07)*, Nara: Springer-Verlag Berlin, 2007, pp. 308–319,

Extension of attack trees using multiple parameters and probabilistic methods. These parameters can be inaccurate or estimated. Parameters include cost, chance of success and extent of attacker penalty. Attackers must have rational benefit.

[86] J. Jürjens, "Model-based Security Testing Using UMLsec. A Case Study," *Electronic Notes in Theoretical Computer Science*, vol. 220, no. 1, pp. 93–104, 2008,

Model based security testing around specifications of systems rather than implementations (which is where most bugs and flaws are based).

[87] Y Kim and I Kim, "Security Issues in Vehicular Networks," in *Proceedings of the 2013 International Conference on Information Networking (ICOIN)*, Bangkok, Thailand: IEEE, 2013, pp. 468–472,

This paper is looks at security in the VANET (Vehicle Ad-Hoc Network context) and discusses threats and attakcs that can be exploited. They conclude with recommendations for possible viable security measures to counter the threat.

[88] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium,*, 2011, pp. 528–533,

Surveys state-of-the-art regarding security in the connected car, with particular emphasis on the intra-vehicular network. The research is categorised into problems with the in-vehicle network and issues with architectural security. This is followed by an exploration of intrusion detection systems, honeypots and threats. The conclusion is that the efforts thus far have been aimed at problem definition rather than security engineering.

[89] R. Klöti, V. Kotronis, and P. Smith, "OpenFlow : A Security Analysis," in *Proceedings of the 21st IEEE International Conference on Network Protocols*, Göttingen, Oct. 2013,

Security analysis and experimental evaluation using STRIDE and attack trees, although context is on Software Designed Networks rather than vehicles.

[90] W. Knowles, A. Baron, and T. Mcgarr, "Analysis and recommendations for standardization in penetration testing and vulnerability assessment," Lancaster University, London, Tech. Rep., 2015. [Online]. Available: http://eprints.lancs.ac.uk/74275/1/Penetration_testing_online_2.pdf (visited on 2016-04-12),

Gives overview of current penetration testing standards and bodies involved either in the testing process or in standardisation.

[91] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41st Design Automation Conference (DAC)*, San Diego, CA: ACM, 2004, p. 753,

Primer for engineers in the challenges of designing secure embedded systems. Solutions are surveyed that could potentially address these problems.

[92] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13-14, pp. 1–38, 2014,

Presents state of the art regarding directed acyclic graphs (of which attack trees are one type). Taxonomises these formalisms. Very useful ground zero paper

[93] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of A Modern Automobile," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, Oakland, CA: IEEE, May 2010, pp. 447–462,

Seminal work on experimental analysis of a vehicle. They were able to affect all parts of the vehicle, including the chassis, powertrain and controls through a wired connection into the CAN bus. Supposedly, they were lambasted for not having any remote attacks, to which the followup is the Checkoway paper.

[94] M. Krasnyansky and M. Holtmann, *Hcitool*, 2002. [Online]. Available: `http://linuxcommand.org/man_pages/hcitool1.htm` (visited on 2017-02-10),

A site that displays the Linux manual page for hcitool and its parameters

[95] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models," in *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh: IEEE Computer Society, 2004, p. 10,

Requirements engineering from a security perspective. Taking a similar perspective to penetration testing in that "anti-models" are constructed (i.e. analogous to precursor attacker models) in order to test against. A form of model based testing.

[96] U. E. Larson and D. K. Nilsson, "Securing vehicles against cyber attacks," Oak Ridge, Tennessee: ACM Press, May 2008, p. 1,

A paper discussing security challenges in the automotive domain

[97] A. Levi, E. Çetintas, M. Aydos, Ç. K. Koç, and M. U. Çaglayan, "Relay Attacks on Bluetooth Authentication and Solutions," in *Proceedings of the 19th International Symposium of Computer and Information Sciences (ISCIS'04)*, Kemer-Antalya: Springer-Verlag Berlin Heidelberg, Oct. 2004, pp. 278–288,

Describes a man-in-the-middle attack by relaying information from one victim device to another. Partial solution is looking at how the delay caused by the relay can be used for detection.

[98] C. Liechti, *PySerial 3.0 Documentation*, 2017. [Online]. Available: `https://pythonhosted.org/pyserial/` (visited on 2017-03-01),

Python module that encapsulates access to the serial port. Required for aftermarket device testing

[99] B Liu, L Shi, Z Cai, and M Li, "Software Vulnerability Discovery Techniques: A Survey," in *Proceedings of the 4th International Conference on Multimedia Information Networking and Security*, Nanjing: IEEE, 2012,

Survey paper on various techniques used to test the security of software to find vulnerabilities. Tools surveyed include those for static analysis, fuzzing, formal verification and penetration testing

[100] Y. Liu and I. Traore, "Systematic security analysis for service-oriented software architectures," in *Proceedings of ICEBE 2007: IEEE International Conference on e-Business Engineering - Workshops: SOAIC 2007; SOSE 2007; SOKM 2007*, 2007, pp. 612–621,

Security risks to software should be managed and deal-with early in the development lifecycle. Many approaches still rely on ad hoc techniques (i.e. subjective prioritisation). Systematic analysis could be used to limit user involvement - in this context using modelling - and can be used for other security perspectives.

[101] D. Lodge, *Hacking the Mitsubishi Outlander PHEV hybrid*, 2016. [Online]. Available: https://www.pentestpartners.com/security-blog/hacking-the-mitsubishi-outlander-phev-hybrid-suv/ (visited on 2017-05-23),

Blog article describing how the WiFi PSK was enumerated, and how this entry point was used to disable the theft alarm on a Mitsubishi Outlander

[102] N. Madenas, A. Tiwari, C. Turner, and S. Peachey, "An analysis of supply chain issues relating to information flow during the automotive product development," *Journal of Manufacturing Technology Management*, vol. 26, no. 8, pp. 1158–1176, 2015,

Explores barriers to information sharing and information flow, and that one of the largest contributing factor is the company culture (for example parameters such as management awareness, perceived risk to the business and security). They look at the automotive industry specifically, although not at cybersecurity (but at the product development phase instead).

[103] A. Marback, H. Do, K. He, S. Kondamarri, and D. Xu, "A threat model-based approach to security testing," *Software - Practice and Experience*, vol. 43, no. 4, pp. 241–258, 2012,

Follows the process of identifying threats through model-based security testing, generating test cases from these models that can be executed with both valid and invalid input. The interesting thing here is that they make use of threat trees (rather than attack trees), which embody potential risk rather than actual attacks. Nevertheless - very useful, if nothing else then to see how MBST is done.

[104] A. Marback, S. Kondamarri, and D. Xu, "Security test generation using threat trees," *2009 ICSE Workshop on Automation of Software Test*, pp. 62–69, 2009,

Earlier work regarding generation of executable test cases from threat trees, to test using model based security testing.

[105] R. Marinescu, M. Saadatmand, A. Bucaioni, C. Seceleanu, and P. Pettersson, "A Model-Based Testing Framework for Automotive Embedded Systems," in *Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, Verona: IEEE, 2014, pp. 38–47,

Another model-based systematic testing framework, specifically for automotive systems, but not from a security perspective (instead based on functional requirements)

[106] R. Martelloni, *Bluesnarfer*, 2012. [Online]. Available: https://github.com/boos/bluesnarfer (visited on 2017-01-23),

Tool that implements the Bluesnarfer attack

[107] S. Mason, "Vehicle remote keyless entry systems and engine immobilisers: Do not believe the insurer that this technology is perfect," *Computer Law & Security Review*, vol. 28, no. 2, pp. 195–200, 2012,

Security in vehicles (concentrating on keyless entry systems) from the perspective of business, insurance and current regulations surrounding this

[108] S. Mauw and M. Oostdijk, "Foundations of Attack Trees," in *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC 2005)*, D. H. Won and S. Kim, Eds., Seoul, South Korea: Springer Berlin Heidelberg, 2005, pp. 186–198,

Descriptive article about attack trees, including its history, formalisms and enhancements since inception of the concept

[109] J. P. Mcdermott, "Attack Net Penetration Testing," in *Proceedings of the 2000 Workshop on New Security Paradigms*, M. E. Zurko and S. J. Greenwald, Eds., Cork: ACM New York, NY, 2000, pp. 15–21,

Conceptualises an attack net, which is a disjunctive Petri net of places representing interesting states of the security relevant bodies in a system. This was extremely interesting, as there is potential for brainstorming (e.g. for penetration testing) and is more descriptive than an attack tree. However, there is no

internal knowledge of the system and so knowledge of any of the states that the attack net places could represent.

[110] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Las Vegas, NV, Tech. Rep., 2015. [Online]. Available: http://illmatics.com/RemoteCarHacking.pdf (visited on 2016-10-24),

The infamous 2015 Jeep hack, through the uConnect network (from a flaw in the WiFi system)

[111] A. Moreno and E. Okamoto, "BlueSnarf Revisited: OBEX FTP Service Directory Traversal," in *International IFIP TC 6 Workshops, PE-CRN, NC-Pro, WCNS, and SUNSET 2011*, vol. 6827, Valencia: Springer Berlin Heidelberg, 2011, pp. 155–166,

Details and demonstrates an alternate form of Bluesnarfing called Bluesnarf++, involving unauthorised directory traversal by sending in the Linux command for "Go to directory above" through the OBEX File Transfer Profile

[112] "Murz", *bluetoothd Permission denied (13) when connecting A2DP headset*, 2016. [Online]. Available: https://bugs.launchpad.net/ubuntu/+source/bluez/+bug/437649 (visited on 2015-10-16),

Bug reported when trying to connect through A2DP

[113] NCC Group, *nOBEX*, 2016. [Online]. Available: https://github.com/nccgroup/nOBEX (visited on 2016-10-10),

Proof of concept tool that takes advantage of Bluetooth synchronisation profiles to fuzz the automotive headunit

[114] National Council of ISACs, *National Council of ISACs: About ISACS*, 2016. [Online]. Available: https://www.nationalisacs.org/about-isacs (visited on 2017-06-01),

Website pertaining to information sharing and analysis centres (ISACs) based in the US

[115] D. K. Nilsson and U. E. Larson, "Secure Firmware Updates over the Air in Intelligent Vehicles," in *Proceedings of IEEE International Conference on Communications Workshops*, Beijing: IEEE, 2008, pp. 380–384,

Review paper discussing the concepts of performing software updates over the air for vehicles, and why it is necessary to do it securely.

[116] D. K. Nilsson and U. E. Larson, "A Defense-in-Depth Approach to Securing the Wireless Vehicle Infrastructure," *Journal of Networks*, vol. 4, no. 7, pp. 552–564, 2009,

Discuss security challenges and possible countermeasures conceptually from a defense-in-depth perspective

[117] S. Nogueira, A. Sampaio, and A. Mota, "Test generation from state based use case models," *Formal Asp. Comput.*, vol. 26, no. 3, pp. 441–490, 2014,

Addresses the automatic generation of test cases from use cases scenarios (including state, input, output and control flows all represented in CSP). Allows for test case selection depending on scenarios, and is mechanised using FDR

[118] Offensive Security, *Kali Linux*, Online. Available at: https://www.kali.org/ [Last accessed: 19-05-2017],

The Kali Linux distribution, which is a distribution that is specifically aimed at security professionals, with associated tools and setups. Based on Debian Linux

[119] D. K. Oka, T. Furue, S. Bayer, and C. Vuillaume, *Analysis of Performing Secure Remote Vehicle Diagnostics*, 2014. [Online]. Available: https://www.escrypt.com/fileadmin/escrypt/pdf/Whitepaper/Analysis_of_Performing_Secure_Remote_Vehicle_Diagnostics.pdf (visited on 2016-02-20),

Explores remote vehicle diagnostics, as traditionally diagnostics are performed by attaching a physical device. Makes some judgement regarding the sets of diagnostics that can be performed remotely, and determines some relevant security properties for each of the diagnostic sets.

[120] D. K. Oka, T. Furue, L. Langenhop, and T. Nishimura, "Survey of Vehicle IoT Bluetooth Devices," in *Proceedings of the 7th International Conference on Service-Oriented Computing and Applications*, Japan: IEEE, 2014, pp. 260–264,

Surveys from publicly available manuals of both cars and Bluetooth devices to see the kinds of Bluetooth implementations there are in a vehicle. Found that the majority of aftermarket devices used fixed easily-guessable PINs. Found that many cars still use legacy pairings, and there were a couple that had fixed

unusable PINs. No information about what these cars might be or how old they might be.

[121] A. L. Opdahl and G. Sindre, "Experimental comparison of attack trees and misuse cases for security threat identification," *Information and Software Technology*, vol. 51, no. 5, pp. 916–932, 2009,

How to include security in the requirements gathering and analysis stage. Industrial takeup has been slow. Experiments include participants taking part in threat identification exercises to measure coverage and perceptions of the techniques used and the tasks at hand. Perception was not correlated with performance.

[122] Penetration Testing Execution Standard, *PTES Technical Guidelines*, 2014. [Online]. Available: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines (visited on 2016-04-10),

Website laying out the technical methodological guidelines for penetration testing. Tends towards network security, but can be generalised as well.

[123] R. C. W. Phan and P. Mingard, "Analyzing the secure simple pairing in bluetooth v4.0," *Wireless Personal Communications*, vol. 64, no. 4, pp. 719–737, 2012,

Formal analysis of Bluetooth version 4.0 SSP pairing association models, including the human intervention part of the pairing mechanism. They demonstrate an attack and compare the association models with each other (but do not take into account Just Works - since that usually doesn't need human input)

[124] B. Potter, "Next Generation Wireless Security Tools," *Network Security*, vol. 2003, no. 9, pp. 4–5, 2003,

A list of possible tools that could be used for wireless hacking. The only one relevant is Bluesniff

[125] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Proc. of the 2007 Future of Software Engineering*, Minneapolis, MN: IEEE Computer Societies, 2007, pp. 55–71,

Brings together current research regarding software engineering in automotive systems, exploring the challenges and constraints on a vehicle, the development and testing lifecycle and the current work done to address these challenges.

[126] Python Software Foundation, *The Python Standard Library*, 2017. [Online]. Available: https://docs.python.org/2/library/index.html (visited on 2016-04-05),

Lists Python 2.7 standard libraries

[127] QNX, *QNX Software Development Platform: Managing User Accounts*, n.d. [Online]. Available: http://www.qnx.com/developers/docs/660/index.jsp?topic=\%252Fcom.qnx.doc.neutrino.user_guide\%252Ftopic\%252Faccounts.html (visited on 2016-01-29),

QNX is an operating system used by the majority of vehicle manufacturers. Some details are given in a manual style regarding management of user accounts in QNX implementations

[128] S Ravi, A Raghunathan, P Kocher, and S Hattagandy, "Security in Embedded Systems: Design Challenges," *ACM Transactions on Embedded Computer Systems*, vol. 3, no. 3, pp. 461–491, 2004,

Early work conceptualising and exploring the design challenges for security in embedded systems. Don't particularly concentrate on automotive, but interesting and useful concepts.

[129] Robert Bosch GmbH, "CAN Specification version 2.0," ROBERT BOSCH GmbH, Stuttgart, Germany, Tech. Rep., 1991. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf (visited on 2017-03-19),

CAN 2.0 specifications and information. The link points to a copy of it

[130] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Computing Surveys*, vol. 35, no. 2, pp. 132–190, 2003,

Interactions between requirements are important to analyse and essential for management. This paper presents a survey on the tools and literature surrounding Requirements Interaction Management (RIM) and substantiate this by discussing seven different research projects.

[131] A. Roscoe, *Understanding Concurrent Systems*, 1st ed. London: Springer-Verlag London, 2010,

Describes the fundamentals of formal methods (particularly using the process algebra CSP) in the context of concurrent systems

[132] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, I. Seskar, R. Ishtiaq, and O. Sangho, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium*, Washington, DC: USENIX Association Berkeley, 2010, p. 21,

Eavesdropping is possible at around 40m using the tyre pressure monitoring system (using a customised software radio attack platform). There are privacy implications, and they recommend some possible countermeasures.

[133] A. Ruddle, D. Ward, B. Weyl, S. Idrees, Y. Roudier, M. Friedewald, T. Leimbach, A. Fuchs, S. Gurgens, O. Henniger, R. Rieke, M. Ritsscher, H. Broberg, L. Apvrille, R. Pacalet, and G. Pedroza, "EVITA Project: Deliverable D2.3 - Security requirements for automotive on-board networks based on dark-side scenarios," Tech. Rep., 2009. [Online]. Available: http://www.evita-project.org/Deliverables/EVITAD2.3.pdf (visited on 2017-05-12),

EVITA deliverable describing requirements gathering for security, using "dark-side scenarios"

[134] SAE International, *SAE J1979 E/E Diagnostic Test Modes*, 2014. [Online]. Available: http://standards.sae.org/j1979_201408/,

Standard that embodies the OBD-II protocol modes and PIDs (which are diagnostic messages) and defines their standard functionalities

[135] ——, *J3061 : Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*, 2016. [Online]. Available: http://standards.sae.org/j3061_201601/ (visited on 2016-02-09),

The first standard concentrating on cybersecurity that is targeted particularly at vehicle and automotive systems.

[136] M. Salfer, H. Schweppe, and C. Eckert, "Efficient Attack Forest Construction for Automotive On-board Networks," in *Proceedings of the 17th International Conference (ISC) on Information Security*, S. S. Chow, J. Camenisch, L. C. Hui, and S. M. Yiu, Eds., Hong Kong: Springer International Publishing, Oct. 2014,

Article looking at the creation of attack "forests" (multitudes of attack trees) formally from an abstract spec of the entire on-board automotive network. They then evaluate efficiency of such a technique

[137] E. Santos, D. Schoop, and A. Simpson, "Formal Models for Automotive Systems and Vehicular Networks : Benefits and Challenges," in *Proceedings of the 2016 IEEE Vehicular Networking Conference (VNC)*, O. Altintas, E. Ekici, M. Tsai, M. Sepulcre, B. Bloessl, and Y.-L. Wei, Eds., Columbus,OH: IEEE, Dec. 2016,

Small study discussing the lack of formal threat models in automotive security testing. Uses Petri nets as a method and applies it to three illustrative examples: engine start without key, odometer value modification, rerouting due to faked warning. No verification takes place.

[138] I. Schieferdecker, J. Grossmann, and M. Schneider, "Model-Based Security Testing," in *Electronic Proceedings in Theoretical Computer Science (EPTCS 80)*, vol. 80, Tallinn, Estonia, Feb. 2012, pp. 1–12,

Provides an extensive survey of model-based security testing techniques, which cover security functional testing, model-based fuzzing, risk and threat-oriented testing and using test patterns

[139] C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, and P. Puschner, "Using SAE J3061 for Automotive Security Requirement Engineering," in *Proceedings of the 2016 Computer Safety, Reliability and Security Workshops: DECSoS*, A. Skavhaug, J. Guiochet, E. Schoitsch, and F. Bitsch, Eds., vol. 9923, Trondheim: Springer, Sep. 2016, pp. 157–170,

Discuss experiences with applying J3061 (concept phase), focusing on Threat Analysis and Risk Assessment. Slightly different phase to that of the thesis tool. High level analysis only. Discusses link with ISO26262.

[140] B. Schneier, *Attack Trees: Modeling Security Threats*, 1999. [Online]. Available: http://www.schneier.com/paper-attacktrees-ddj-ft.html (visited on 2016-07-10),

Seminal article conceptualising the attack tree, and describing how it can be used to model security threats.

[141] Sparkfun Electronics, *ELM327 AT Commands*, 2010. [Online]. Available: https://www.sparkfun.com/datasheets/Widgets/ELM327_AT_Commands.pdf (visited on 2017-05-12),

List of the set of AT commands supported by various versions of the ELM chip

[142] T. Sridhar, "Wi-Fi, Bluetooth and WiMax - Technology and Implementation," *The Internet Protocol Journal*, vol. 11, no. 4, pp. 2–17, 2008,

A general article explaining the workings of Bluetooth and other wireless protocols

[143] R. I. of Sweden, *Heavens*, 2017. [Online]. Available: https://www.sp.se/en/index/research/dependable_systems/heavens/sidor/default.aspx (visited on 2017-07-13),

Online home of the HEAVENs project. Now completed, but only very brief information given

[144] SyncML, "SyncML OBEX Binding," Open Mobile Alliance, Tech. Rep., 2001, pp. 1–19. [Online]. Available: http://technical.openmobilealliance.org/tech/affiliates/syncml/syncml_obex_v101_20010615.pdf (visited on 2017-01-22),

SyncML specifications and how it binds to OBEX in Bluetooth

[145] The MITRE Corporation, *Common Vulnerabilities and Exposures (CVE) List*, 2017. [Online]. Available: https://cve.mitre.org/ (visited on 2017-01-03),

Website containing a database of publicly known vulnerabilities

[146] ——, *Common Weakness Enumeration*, 2017. [Online]. Available: https://cwe.mitre.org/ (visited on 2017-05-20),

A database containing details of software weakness types

[147] H. H. Thompson, "Why security testing is hard," *IEEE Security and Privacy*, vol. 1, no. 4, pp. 83–86, 2003,

Position paper talking about the development of techniques, methods nad tools dealing with software flaws

[148] P. Torr, "Demystifying the threat modeling process," *IEEE Security & Privacy Magazine*, vol. 3, pp. 66–70, 2005,

Considers security needs during the design phase. Looks at threat modelling from a general perspective, and explores a little bit of the STRIDE methodology

[149] Trifinite Group, *Bluesmack*, n.d. [Online]. Available: http://trifinite.org/trifinite_stuff_bluesmack.html (visited on 2015-10-09),

An attack that causes denial of service to susceptible devices by flooding with large L2CAP messages

[150] University of Oxford, *FDR3 - The CSP Refinement Checker*, 2016. [Online]. Available: https://www.cs.ox.ac.uk/projects/fdr/ (visited on 2016-05-10),

A tool developed by the University of Oxford for refinement checking of statements made in the process algebra CSP. It allows CSP processes to be defined as machine-readable processes (CSP_M), which it can then check against various assertions.

[151] C. Vallance, *Car hack uses digital-radio broadcasts to seize control*, 2015. [Online]. Available: http://www.bbc.co.uk/news/technology-33622298,

News report detailing the efforts of the NCC Group who have managed to use DAB radio to take control of a vehicle

[152] R Verdult, F. Garcia, and J. Balasch, "Gone in 360 seconds: Hijacking with Hitag2," in *Proceedings of 21st USENIX conference on Security symposium*, Bellevue, WA: USENIX Association Berkeley, Aug. 2012, p. 37. (visited on 2014-12-11),

Motivation is the theft of vehicles. The Hitag2 chip is used in keys and this paper demonstrates how the encryption on this can be broken (as it only uses 48-bit keys) and how an emulating transponder could be used to start engines without the legitimate keys

[153] R. Verdult, F. D. Garcia, and B. Ege, "Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer," in *Supplement to the Proceedings of the 22nd USENIX Security Symposium*, Washington, DC: USENIX Association, 2013,

Paper presenting the breakdown of how Megamos (a cryptographic protocol used in key-vehicle exchange to disengage the immobiliser) can be broken. This paper is infamous for having an injunction filed against it by a major vehicle manufacturer. The embargo on publication was lifted in 2015.

[154] R. Vigo, F. Nielson, and H. R. Nielson, "Automated generation of attack trees," in *Proc. of the 27th Computer Security Foundations Symposium*, Vienna: IEEE, 2014,

Automatic inference of attack trees based on a process algebra specification which would allow for solving quantitative problems as well as addresses scalability. Demonstrated using a case study of a national-scale authentication system

[155] C. Weissman, "Penetration Testing (Handbook for the Computer Security Certifications of Trusted Systems)," Naval Research Laboratory, Washington, DC, Tech. Rep., 1995,

Early written work regarding the process of penetration testing

[156] O. Whitehouse, S. Halsall, and S. Kapp, *Redfang*, 2003. [Online]. Available: http://tools.kali.org/wireless-attacks/redfang (visited on 2017-04-19),

Small proof-of-concept tool for brute-scanning for hidden devices by cycling through addresses and incrementing by one each time. Incorporated into Kali Linux

[157] J. A. Whittaker, "What is software testing? And why is it so hard?" *IEEE Software*, vol. 17, no. 1, pp. 70–79, 2000,

Explores and discusses the challenges in software testing, including non-documentation, non-systematism and how to address these.

[158] M. Wolf and M. Scheibel, "A Systematic Approach to a Quantified Security Risk Analysis for Vehicular IT Systems," in *ESCRYPT Automotive Safety and Security*, Karlsruhe: ESCRYPT, 2012, pp. 195–210,

Proposes a systematic approach for a full and quantifiable risk analysis, to avoid over and under protection of systems. Applies sytematic estimations for risk analysis, using potential damages and probability of a successful security attack, adapted to vehicular IT security scenarios. Related and overlaps a little

with the work done in this thesis, except for the fact that they are looking at risk rather than security testing.

[159] M. Wolf, A. Weimerskirch, and C. Paar, "Security in Automotive Bus Systems," in *Proceedings of the Workshop on Embedded Security in Cars (ESCAR)*, 2004, pp. 1–13,

Review paper describing reported attacks on automotive intra-vehicular networks, as well as possible countermeasures

[160] M. Wolf, A. Weimerskirch, and T. Wollinger, "State of the art: Embedding security in vehicles," *EURASIP Journal on Embedded Systems*, vol. 2007:074706, 2007,

Review paper describing the processes underway to embed security in vehicles as opposed to retro-fitting

[161] S. Woo, H. J. Jo, and D. H. Lee, "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN," *IEEE Transactions on Intelligent Transport Systems*, vol. 16, no. 2, pp. 993–1006, 2015,

Discusses the melding of automotive and IT technology, which is dangerous to the vehicle. Security was not considered in CAN and the vehicle is now opening up to external networks. They show that a long-range wireless attack is possible using a real vehicle and a malicious smartphone app. They then propose a security protocol for CAN and evaluate feasibility. Conclusion: that the proposed security protocol is more efficient than other proposals, especially with regards to authentication delay and network load.

[162] J. Wright and J. Cache, "Bluetooth Basics (Extended Web Edition)," in *Hacking Exposed Wireless: Wireless Security Secrets & Solutions*, 3rd, USA: McGraw-Hill Education, 2015, ch. 2, pp. 1–32. [Online]. Available: http://www.hackingexposedwireless.com/chapters/ch02.pdf (visited on 2015-12-20),

Explains the basics of Bluetooth, as well as presents some examples of working with Bluetooth through Python

[163] D. Xu and K. E. Nygard, "Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets," *IEEE Transactions on Software Engineering*, vol. 32, no. 4, pp. 265–278, 2006,

Formal modeling and verification of software using petri nets, which is a type of model used for concurrent and distributed systems and allows for reasoning

[164] K. P. Yee, "User interaction design for secure systems," in *Proceedings of the 4th International Conference on Information and Communications Security (ICICS)*, Berkeley: Springer, Dec. 2002, pp. 278–290,

Looks at how the security of a system that is human dependent, and the effect of the user interface, user decision and interpretation of user actions on this property. Work mostly explores reasoning behind user-centred security, in terms of actors and actions. Finishes off with real-world case studies.

[165] N. Yegorov, *Lightblue*, 2014. [Online]. Available: https://github.com/0-1-0/lightblue-0.4 (visited on 2016-05-04),

Alternate cross-platform Python Bluetooth library

[166] karulis, *Pybluez: Bluetooth Python extension module*, 2017. [Online]. Available: https://github.com/karulis/pybluez (visited on 2016-11-10),

Python wrapper for Bluetooth functionality on Linux

[167] nu: The Open Security Community, *Carwhisperer, Bluetooth Attack*, 2012. [Online]. Available: http://www.slideshare.net/null0x00/carwhisperer-bluetooth-attack (visited on 2015-10-14),

Presentation detailing the CarWhisperer attack, which involves trying out default PINs of oooo and 1234 to gain access and then using the SCO protocol to inject audio files.

Part IV

SUPPLEMENTARY MATERIALS

A

APPENDIX

A.1 REQUIRED PYTHON LIBRARIES

The below lists the versions of the Python libraries used and is the minimum version required for the proof-of-concept tool. A description is also given of the functionality each of the libraries provides the tool [126].

Library	*Version*	*Description*
Python Standard Libraries [126]		
re	Python 2.7	Provides regular expression operations, used for input validation
time	Python 2.7	Provides ability to specify time intervals for input
subprocess	Python 2.7	Allows for spawning of new processes
shutil	Python 2.7	Provides operations for high level file manipulation, used for manipulating created logs
os	Python 2.7	Provides access to miscellaneous operating system interfaces, used for file object creation
csv	Python 2.7	Used to read and write to Comma-Separated Values format files, used for manipulating created logs
stringIO	Python 2.7	Provides ability to read and write strings as files, used for logging functions
Bluetooth-related libraries		
PyBluez	0.22	Python extension library providing access to system Bluetooth resources [166]
bluetool	0.2.1	Python API for Bluetooth D-Bus calls, used for pairing checks [3]
lightblue	0.4	Python library providing access to Bluetooth operations [165]
python-obexftp	0.26	Python implementation for aspects of the OBEX protocol [25], used for File Transfer Profile tests
pySerial	3.0	Python module encapsulating access to a serial port [98], required for access to Bluetooth's Serial Port Profile
Other libraries		
treelib	1.3.5	Tree data structure implementation in Python [37], required to build, display and search tree structure
libfdr	4.2.0	FDR API (64-bit only) which exposes part of FDR's internals for external tool use
PTable	0.9.2	Python library to represent tabular data [16], used for tabulating diagnosis results
PyFiglet	0.7.5	Creates ASCII art [84]

A.2 DOMAIN EXPERT REVIEWER BIOGRAPHIES

Paul Wooderson is a Senior Functional Safety and Cybersecurity Engineer at HORIBA MIRA Ltd, responsible for cybersecurity research and development. He is a Chartered Engineer with 15 years' experience in embedded systems security in the automotive and smartcard domains. Paul's experience includes threat analysis and risk assessment, security evaluation of cryptographic hardware and software, secure design techniques and security certification. Paul is also a UK Expert to the joint ISO/SAE working group on automotive cybersecurity engineering and a member of the SAE Vehicle Cybersecurity Systems Engineering Committee.

Anthony Jude is a Senior Systems and Cyber Security Engineer at HORIBA MIRA Ltd., with expertise in weapon systems, electronic warfare, radio frequency communications and substation automation. He has 12 years of experience in developing software centric systems, five years in developing mechatronic systems to IEC61508 and two years experience developing safety-related mechatronic systems to ISO26262.

B

ETHICS DOCUMENTATION

Low Risk Research Ethics Approval

Project Title

Towards a systematic security evaluation of the automotive Bluetooth interface

Record of Approval

Principal Investigator

I **request an ethics peer review** and confirm that I have answered all relevant questions in this checklist honestly.	X
I confirm that I will carry out the project in the ways described in this checklist. I will immediately suspend research and request new ethical approval if the project subsequently changes the information I have given in this checklist.	X
I confirm that I, and all members of my research team (if any), have read and agreed to abide by the Code of Research Ethics issued by the relevant national learned society.	X
I confirm that I, and all members of my research team (if any), have read and agreed to abide by the University's Research Ethics, Governance and Integrity Framework.	X

Name: Hun Cheah .....

Date: 06/07/2016 .....

Student's Supervisor (if applicable)

I have read this checklist and confirm that it covers all the ethical issues raised by this project fully and frankly. I also confirm that these issues have been discussed with the student and will continue to be reviewed in the course of supervision.

Name: Siraj Shaikh .....

Date: 09/05/2017 .....

Reviewer (if applicable)

Date of approval by anonymous reviewer: 19/06/2017

Low Risk Research Ethics Approval Checklist

Project Information

Project Ref	P45211
Full name	Hun Cheah
Faculty	Faculty of Engineering, Environment and Computing
Department	School of Computing, Electronics and Maths
Supervisor	Siraj Shaikh
Module Code	ECCOM
EFAAF Number	
Project title	Towards a systematic security evaluation of the automotive Bluetooth interface
Date(s)	07/07/2016 - 01/09/2017
Created	06/07/2016 17:38

Project Summary

The modern vehicle requires connectivity in order to enable and enhance comfort and convenience features so desired by customers. This connectivity however also allows the possibility that an external attacker may compromise the security (and therefore the safety) of the vehicle. In order to answer this problem, we propose a framework for a systematic method of security testing for automotive Bluetooth interfaces and implement a proof-of-concept tool to carry out testing on vehicles using this framework.

Names of Co-Investigators and their organisational affiliation (place of study/employer)	
Is the project self-funded?	YES
Who is funding the project?	Coventry University
Has the funding been confirmed?	YES
Are you required to use a Professional Code of Ethical Practice appropriate to your discipline?	NO
Have you read the Code?	NO

Project Details

What is the purpose of the project?	To test vehicles with the developed proof-of-concept tool in order to enumerate weaknesses
What are the planned or desired outcomes?	That the structured systematic testing framework will find weaknesses in Bluetooth implementations in vehicles
Explain your research design	The approach is based on attack trees and penetration testing methods, which in turn leads to the development of a semi-automated tool in order to test vehicles.
Outline the principal methods you will use	Reconnaissance and information gathering to find vehicle and device information, along with penetration testing methods to identify or determine points of weaknesses in the automotive implementation of Bluetooth
Are you proposing to use an external research instrument, validated scale or follow a published research method?	NO
If yes, please give details of what you are using	
Will your research involve consulting individuals who support, or literature, websites or similar material which advocates, any of the following: terrorism, armed struggles, or political, religious or other forms of activism considered illegal under UK law?	NO
Are you dealing with Secondary Data? (e.g. sourcing info from websites, historical documents)	YES
Are you dealing with Primary Data involving people? (e.g. interviews, questionnaires, observations)	NO
Are you dealing with personal or sensitive data?	NO
Is the project solely desk based? (e.g. involving no laboratory, workshop or off-campus work or other activities which pose significant risks to researchers or participants)	NO
Are there any other ethical issues or risks of harm raised by the study that have not been covered by previous questions?	NO
If yes, please give further details	

External Ethical Review

Question		Yes	No
1	Will this study be submitted for ethical review to an external organisation? (e.g. Another University, Social Care, National Health Service, Ministry of Defence, Police Service and Probation Office)		X
	If YES, name of external organisation		
2	Will this study be reviewed using the IRAS system?		X
3	Has this study previously been reviewed by an external organisation?		X

Risk of harm, potential harm and disclosure of harm

Question			Yes	No
1	Is there any significant risk that the study may lead to physical harm to participants or researchers?			X
	If YES, please explain how you will take steps to reduce or address those risks			
2	Is there any significant risk that the study may lead to psychological or emotional distress to participants?			X
	If YES, please explain how you will take steps to reduce or address those risks			
3	Is there any risk that the study may lead to psychological or emotional distress to researchers?			X
	If YES, please explain how you will take steps to reduce or address those risks			
4	Is there any risk that your study may lead or result in harm to the reputation of participants, researchers, or their employees, or any associated persons or organisations?			X
	If YES, please explain how you will take steps to reduce or address those risks			
5	Is there a risk that the study will lead to participants to disclose evidence of previous criminal offences, or their intention to commit criminal offences?			X
	If YES, please explain how you will take steps to reduce or address those risks			
6	Is there a risk that the study will lead participants to disclose evidence that children or vulnerable adults are being harmed, or at risk or harm?			X
	If YES, please explain how you will take steps to reduce or address those risks			
7	Is there a risk that the study will lead participants to disclose evidence of serious risk of other types of harm?			X
	If YES, please explain how you will take steps to reduce or address those risks			
8	Are you aware of the CU Disclosure protocol?		X	

Online and Internet Research

Question			Yes	No
1	Will any part of your study involve collecting data by means of electronic media (e.g. the Internet, e-mail, Facebook, Twitter, online forums, etc)?			X
	If YES, please explain how you will obtain permission to collect data by this means			
2	Is there a possibility that the study will encourage children under 18 to access inappropriate websites, or correspond with people who pose risk of harm?			X
	If YES, please explain further			
3	Will the study incur any other risks that arise specifically from the use of electronic media?			X
	If YES, please explain further			
4	Will you be using survey collection software (e.g. BoS, Filemaker)?			X
	If YES, please explain which software			
5	Have you taken necessary precautions for secure data management, in accordance with data protection and CU Policy?		X	
	If NO	please explain why not		
	If YES	Specify location where data will be stored	Either on University computer or on a computer (and network space) provided by collaborating organisation.	
		Planned disposal date	01/09/2018	
		If the research is funded by an external organisation, are there any requirements for storage and disposal?		X
		If YES, please specify details		

Laboratory/Workshops

Question			Yes	No
1	Does any part of the project involve work in a laboratory or workshop which could pose risks to you, researchers or others?			X
	If YES: If you have risk assessments for laboratory or workshop activities you can refer to them here & upload them at the end, or explain in the text box how you will manage those risks			

Research with non-human vertebrates

Question		Yes	No
1	Will any part of the project involve animal habitats or tissues or non-human vertebrates?		X
	If YES, please give details		
2	Does the project involve any procedure to the protected animal whilst it is still alive?		
3	Will any part of your project involve the study of animals in their natural habitat?		
	If YES, please give details		
4	Will the project involve the recording of behaviour of animals in a non-natural setting that is outside the control of the researcher?		
	If YES, please give details		
5	Will your field work involve any direct intervention other than recording the behaviour of the animals available for observation?		
	If YES, please give details		
6	Is the species you plan to research endangered, locally rare or part of a sensitive ecosystem protected by legislation?		
	If YES, please give details		
7	Is there any significant possibility that the welfare of the target species of those sharing the local environment/habitat will be detrimentally affected?		
	If YES, please give details		
8	Is there any significant possibility that the habitat of the animals will be damaged by the project, such that their health and survival will be endangered?		
	If YES, please give details		
9	Will project work involve intervention work in a non-natural setting in relation to invertebrate species other than Octopus vulgaris?		
	If YES, please give details		

Blood Sampling / Human Tissue Analysis

Question			Yes	No
1	Does your study involve collecting or use of human tissues or fluids? (e.g. collecting urine, saliva, blood or use of cell lines, 'dead' blood)			X
	If YES, please give details			
2	If your study involves blood samples or body fluids (e.g. urine, saliva) have you clearly stated in your application that appropriate guidelines are to be followed (e.g. The British Association of Sport and Exercise Science Physiological Testing Guidelines (2007) or equivalent) and that they are in line with the level of risk?			
	If NO, please explain why not			
3	If your study involves human tissue other than blood and saliva, have you clearly stated in your application that appropriate guidelines are to be followed (e.g. The Human Tissues Act, or equivalent) and that they are in line with level of risk?			
	If NO, please explain why not			

Travel

Question			Yes	No
1	Does any part of the project require data collection off campus? (e.g. work in the field or community)			X
	If YES: You must consider the potential hazards from off campus activities (e.g. working alone, time of data collection, unfamiliar or hazardous locations, using equipment, the terrain, violence or aggression from others). Outline the precautions that will be taken to manage these risks, AS A MINIMUM this must detail how researchers would summon assistance in an emergency when working off campus. For complex or high risk projects you may wish to complete and upload a separate risk assessment			
2	Does any part of the project involve the researcher travelling outside the UK (or to very remote UK locations)?			
	If YES: Please give details of where, when and how you will be travelling. For travel to high risk places you may wish to complete and upload a separate risk assessment			
3	Are all travellers aware of contact numbers for emergency assitance when away (e.g. local emergency assistance, ambulance/local hospital/police, insurance helpline [+44 (0) 2071 737797] and CU's 24/7 emergency line [+44 (0) 2476 888555])?			
4	Are there any travel warnings in place advising against all, or essential only travel to the destination? NOTE: Before travel to countries with 'against all travel', or 'essential only' travel warnings, staff must check with Finance to ensure insurance coverage is not affected. Undergraduate projects in high risk destinations will not be approved			
5	Are there increased risks to health and safety related to the destination? e.g. cultural differences, civil unrest, climate, crime, health outbreaks/concerns, and travel arrangements?			
	If YES, please specify			
6	Do all travelling members of the research team have adequate travel insurance?			
7	Please confirm all travelling researchers have been advised to seek medical advice regarding vaccinations, medical conditions etc, from their GP			

Certificate of Ethical Approval

Student:

Hun Cheah

Project Title:

Vehicular Cyber-security

This is to certify that the above named student has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

11 January 2015

Project Reference Number:

P29187

REGISTRY RESEARCH UNIT

ETHICS REVIEW FEEDBACK FORM

(Review feedback should be completed within 10 working days)

Name of applicant: Hun Cheah .....

Faculty/School/Department: [Faculty of Engineering and Computing] Computing

Research project title: Vehicular Cyber-security

Comments by the reviewer

1. Evaluation of the ethics of the proposal:

2. Evaluation of the participant information sheet and consent form:

3. Recommendation:

(Please indicate as appropriate and advise on any conditions. If there any conditions, the applicant will be required to resubmit his/her application and this will be sent to the same reviewer).

	Approved - no conditions attached
	Approved with minor conditions (no need to re-submit)
	Conditional upon the following – please use additional sheets if necessary (please re-submit application)
	Rejected for the following reason(s) – please use other side if necessary
X	Not required

Name of reviewer: Anonymous.....

Date: 11/01/2015.....

Certificate of Ethical Approval

Applicant:

Hun Cheah

Project Title:

Security testing of automotive wireless communication interfaces

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

27 July 2015

Project Reference Number:

P35838

Low Risk Research Ethics Approval

Project Title

Security testing of automotive wireless communication interfaces

Record of Approval

Principal Investigator

I **request an ethics peer review** and confirm that I have answered all relevant questions in this checklist honestly.	X
I confirm that I will carry out the project in the ways described in this checklist. I will immediately suspend research and request new ethical approval if the project subsequently changes the information I have given in this checklist.	X
I confirm that I, and all members of my research team (if any), have read and agreed to abide by the Code of Research Ethics issued by the relevant national learned society.	X
I confirm that I, and all members of my research team (if any), have read and agreed to abide by the University's Research Ethics, Governance and Integrity Framework.	X

Name: Hun Cheah.....

Date: 27/07/2015.....

Student's Supervisor (if applicable)

I have read this checklist and confirm that it covers all the ethical issues raised by this project fully and frankly. I also confirm that these issues have been discussed with the student and will continue to be reviewed in the course of supervision.

Name: Siraj Shaikh.....

Date: 27/07/2015.....

Reviewer (if applicable)

Date of approval by anonymous reviewer: 27/07/2015

Low Risk Research Ethics Approval Checklist

Project Information

Project Ref	P35838
Full name	Hun Cheah
Faculty	Faculty of Engineering and Computing
Department	Computing
Supervisor	Siraj Shaikh
Module Code	ECCOM
EFAAF Number	
Project title	Security testing of automotive wireless communication interfaces
Date(s)	29/07/2015 - 29/07/2016
Created	27/07/2015 14:51

Project Summary

The purpose of the project is to develop a framework for the assessment of how secure a vehicle (and the connections it makes) is, and how secure it should be. This is to be achieved in different approaches: A black box penetration testing kit to be used on a production vehicle provided by MIRA A spectrum analysis of any communication thereof using open source tools

Names of Co-Investigators and their organisational affiliation (place of study/employer)	
Is the project self-funded?	YES
Who is funding the project?	Coventry University and HORIBA MIRA Ltd.
Has the funding been confirmed?	YES
Are you required to use a Professional Code of Ethical Practice appropriate to your discipline?	YES
Have you read the Code?	YES

Project Details

What is the purpose of the project?	The purpose is to ascertain the level of security of wireless communications of a vehicle, specifically, in this case Bluetooth
What are the planned or desired outcomes?	The planned outcome is a set of results that can be used to ascertain and identify: pertinent security test cases coverage of test cases testing of cases that could be done within constraints and how this might affect the overall security landscape of the vehicle
Explain your research design	Using open source tools in order to test the various aspects of the Bluetooth connections on a production vehicle. This includes the use of the Penetration Testing Execution Standards framework, following the steps of: - information gathering - enumeration - footprinting - exploitation
Outline the principal methods you will use	Information gathering involves the use of public data to ascertain as much general information as possible on the test bed. Enumeration involves looking at the services offered and the channels open through which testing can take place. Footprinting, whilst more invasive will be looking at detailing exactly the interface to be tested and the behind-the-scenes operation. Exploitation will involve the use of various tools to compromise mechanisms (if there are any) present within this interface to the vehicle.
Are you proposing to use an external research instrument, validated scale or follow a published research method?	NO
If yes, please give details of what you are using	
Will your research involve consulting individuals who support, or literature, websites or similar material which advocates, any of the following: terrorism, armed	NO

struggles, or political, religious or other forms of activism considered illegal under UK law?	
Are you dealing with Secondary Data? (e.g. sourcing info from websites, historical documents)	NO
Are you dealing with Primary Data involving people? (e.g. interviews, questionnaires, observations)	NO
Are you dealing with personal or sensitive data?	NO
Is the project solely desk based? (e.g. involving no laboratory, workshop or off-campus work or other activities which pose significant risks to researchers or participants)	NO
Are there any other ethical issues or risks of harm raised by the study that have not been covered by previous questions?	NO
If yes, please give further details	

Laboratory/Workshops

Question			Yes	No
1	Does any part of the project involve work in a laboratory or workshop which could pose risks to you, researchers or others?			X
	If YES: If you have risk assessments for laboratory or workshop activities you can refer to them here & upload them at the end, or explain in the text box how you will manage those risks			

Research with non-human vertebrates

Question			Yes	No
1	Will any part of the project involve animal habitats or tissues or non-human vertebrates?			X
	If YES, please give details			
2	Does the project involve any procedure to the protected animal whilst it is still alive?			
3	Will any part of your project involve the study of animals in their natural habitat?			
	If YES, please give details			
4	Will the project involve the recording of behaviour of animals in a non-natural setting that is outside the control of the researcher?			
	If YES, please give details			
5	Will your field work involve any direct intervention other than recording the behaviour of the animals available for observation?			
	If YES, please give details			
6	Is the species you plan to research endangered, locally rare or part of a sensitive ecosystem protected by legislation?			
	If YES, please give details			
7	Is there any significant possibility that the welfare of the target species of those sharing the local environment/habitat will be detrimentally affected?			
	If YES, please give details			
8	Is there any significant possibility that the habitat of the animals will be damaged by the project, such that their health and survival will be endangered?			
	If YES, please give details			
9	Will project work involve intervention work in a non-natural setting in relation to invertebrate species other than Octopus vulgaris?			
	If YES, please give details			

Blood Sampling / Human Tissue Analysis

Question			Yes	No
1	Does your study involve collecting or use of human tissues or fluids? (e.g. collecting urine, saliva, blood or use of cell lines, 'dead' blood)			X
	If YES, please give details			
2	If your study involves blood samples or body fluids (e.g. urine, saliva) have you clearly stated in your application that appropriate guidelines are to be followed (e.g. The British Association of Sport and Exercise Science Physiological Testing Guidelines (2007) or equivalent) and that they are in line with the level of risk?			
	If NO, please explain why not			
3	If your study involves human tissue other than blood and saliva, have you clearly stated in your application that appropriate guidelines are to be followed (e.g. The Human Tissues Act, or equivalent) and that they are in line with level of risk?			
	If NO, please explain why not			

Travel

Question			Yes	No
1	Does any part of the project require data collection off campus? (e.g. work in the field or community)			X
	If YES: You must consider the potential hazards from off campus activities (e.g. working alone, time of data collection, unfamiliar or hazardous locations, using equipment, the terrain, violence or aggression from others). Outline the precautions that will be taken to manage these risks, AS A MINIMUM this must detail how researchers would summon assistance in an emergency when working off campus. For complex or high risk projects you may wish to complete and upload a separate risk assessment			
2	Does any part of the project involve the researcher travelling outside the UK (or to very remote UK locations)?			
	If YES: Please give details of where, when and how you will be travelling. For travel to high risk places you may wish to complete and upload a separate risk assessment			
3	Are all travellers aware of contact numbers for emergency assitance when away (e.g. local emergency assistance, ambulance/local hospital/police, insurance helpline [+44 (0) 2071 737797] and CU's 24/7 emergency line [+44 (0) 2476 888555])?			
4	Are there any travel warnings in place advising against all, or essential only travel to the destination? NOTE: Before travel to countries with 'against all travel', or 'essential only' travel warnings, staff must check with Finance to ensure insurance coverage is not affected. Undergraduate projects in high risk destinations will not be approved			
5	Are there increased risks to health and safety related to the destination? e.g. cultural differences, civil unrest, climate, crime, health outbreaks/concerns, and travel arrangements?			
	If YES, please specify			
6	Do all travelling members of the research team have adequate travel insurance?			
7	Please confirm all travelling researchers have been advised to seek medical advice regarding vaccinations, medical conditions etc, from their GP			